UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Desarrollo de un sistema de detección de tráfico anómalo en el contexto de la Internet de las Cosas

David Olano Arias

Tutor: Jorge E. López de Vergara Méndez

FEBRERO 2016

Desarrollo de un sistema de detección de tráfico anómalo en el contexto de la Internet de las Cosas

AUTOR: David Olano Arias

TUTOR: Jorge E. López de Vergara Méndez

High Performance Computing and Networking Research Group

Dpto. de Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Febrero de 2016

Resumen

La Internet de las Cosas (*IoT*) es un nuevo paradigma de comunicación que ha ganado peso en los últimos años en el ámbito de la red. Este concepto, propuesto por Kevin Ashton en el año 1999, consiste en la idea básica de interconectar los objetos cotidianos que nos rodean con Internet a través de dispositivos inalámbricos como las etiquetas RFID o sistemas NFC, generando una malla de *"cosas"* con capacidad de comunicarse entre sí y con Internet.

La unión del mundo virtual con el mundo físico podría suponer un cambio total en la manera en que entendemos la red. Dicha interconexión permitirá adquirir datos de nuestro entorno de forma automática y sistematizada, controlar objetos domésticos de manera remota y supondrá, entre otros avances, un salto tecnológico sustancial en el campo de la telemedicina, la domótica, las *Smart Grids* y el *e-learning*.

Una implementación práctica de IoT tiene ciertas similitudes con un sistema industrial de control de procesos en tanto en cuanto existen objetos que funcionan como sensores y actuadores que interactúan entre sí y con los clientes para alcanzar un determinado objetivo. Es por ello que se tiende a plantear el uso de sistemas SCADA provenientes de la industria para modelar este tipo de comunicaciones.

Uno de los protocolos SCADA de mayor importancia es DNP3 (*Distributed Network Protocol 3)*, el cual fue desarrollado por la empresa canadiense GE-Harris en 1993 con la premisa de ser un sistema de comunicación muy fiable incluso en condiciones desfavorables de ruido electromagnético, temperatura y medios de transmisión ineficaces. Este protocolo, si bien altamente fiable, no fue diseñado para ser especialmente seguro dadas las limitaciones de procesamiento, memoria y ancho de banda que existían en los años noventa.

En el contexto de la interconexión de la Internet de las Cosas la ciberseguridad reviste capital importancia, ya que un ataque realizado con éxito sobre un sistema de estas características tendría efecto sobre los objetos físicos además del impacto sobre el sistema virtual. Éste hecho, unido al paulatino aumento de los ciberataques y la evolución de las técnicas de *hacking* sobre infraestructuras críticas durante la última década, implican la obligatoriedad de mejorar los sistemas de ciber-defensa para asegurar los principios básicos de confidencialidad, integridad y disponibilidad.

Debido a que el protocolo DNP3 no provee por sí mismo de mecanismos de seguridad efectivos y que el aislamiento de los sistemas como medida de seguridad no es una opción en una aplicación de estas características, se hace necesario valerse de sistemas adicionales que permitan compensar estas carencia, como sistemas de detección de intrusiones adaptados a los nuevos usos del protocolo.

Por todo lo anterior, este Trabajo Fin de Grado estudia y trata de aplicar los medios de defensa necesarios para avalar un nivel de protección aceptable en una red IoT que se base en dicho protocolo DNP3.

Palabras clave

Internet de las Cosas, DNP3, SCADA, monitorización, detección, Python, IDS

Abstract

The Internet of Things (*IoT)* is a new communication paradigm that has gained attention in recent years in the network sphere. This concept, proposed by Kevin Ashton in the year 1999, consists of the basic idea of creating an interconnection among daily objects and the Internet through wireless devices such as RFID tags o NFC systems; creating a network of "*things*" that can communicate with each other and with the Internet.

The connection between the virtual world and the physical world could suppose a radical change in the way in which we understand the network. That interconnection will allow obtaining data from our environment in an automatic and systematic way, to control domestic objects through a remote connection and, furthermore, it will suppose a technological forward leap in the field of telemedicine, home automation, smart grids and e-learning.

A practical implementation of the IoT has some similarities with a process control in an industrial system. Both systems have objects that work as sensors and actuators that interact with each other, and with the clients, in order to achieve a certain objective. That is the main reason why it is common to think on SCADA systems stemming from the industry to shape this kind of communications.

One of the most important SCADA protocols is the DNP3 (Distributed Network Protocol 3), which was created by the Canadian company GE-Harris in 1993 based on the not-yet-standardized specifications of the IEC 60870-5. This protocol was created under the premise of being a highly reliable communication system, even under unfavourable conditions such as electromagnetic interferences, unfavourable temperature and inefficient modes of transmission. Despite being a highly reliable protocol, it was not designed to be especially secure, bearing in mind the processing, memory and bandwidth limitations in the nineties.

Cybersecurity is a very important element in the interconnection context. A successful attack in a system of such characteristics could have an effect on the physical objects and the virtual system itself. This fact, together with the gradual increase of cyberattacks and the progression of the hacking techniques affecting critical infrastructures over the last decade, makes the upgrading of cyber defence systems a priority in order to assure the basic principles of confidentiality, integrity and availability.

Due to the lack of security mechanisms in the DNP3 protocol, and bearing in mind that the isolation of the systems as a security measure is not a valid option in an application of such characteristics, it is necessary to establish additional systems, such as intrusion detection systems adapted to the new uses of the protocol, to compensate this scarcity.

For all the above reasons, this dissertation studies and tries to apply the necessary defence methods to obtain a valid level of protection in an IoT network based on DNP3 protocol.

Index terms

Internet of things, DNP3, SCADA, monitoring, detection, Python, IDS

Agradecimientos

Este Trabajo Fin de Grado supone el punto y final de la aventura que he vivido durante los últimos cinco años. Echando la vista atrás, surgen un sinfín de buenos momentos, experiencias y anécdotas, por lo que resulta complicado agradecer en esta página a todas las personas que han tenido algo que ver.

En primer lugar, a las personas que han estado y están más cerca de mí; mi madre, Mª Ángeles; mi padre, Tomás; mi hermana, Sara y mi pareja, Izar. Gracias por vuestro apoyo incondicional y por haber creído que llegar hasta aquí era posible incluso cuando ni yo mismo lo hacía. Esto es más vuestro que mío.

Al tutor de este Trabajo Fin de Grado, Jorge López de Vergara, por la dedicación y disponibilidad mostradas durante el desarrollo del proyecto. Tu apoyo, cercanía y capacidad para la resolución de problemas han sido determinantes para conseguir llevar el trabajo a buen puerto.

A Álvaro Culebras, ya que sin su ayuda y experiencia los comienzos del proyecto hubieran sido mucho más duros.

A Nicholas Rodofile, doctorando de la *Queensland University of Technology*, por su ayuda con las bibliotecas de DNP3 para Scapy. También a José Ángel Abuín, por sus consejos sobre el entorno de trabajo y a Sergio Valeriano, cuyo dominio del inglés me ha facilitado mucho la redacción de la memoria.

Por último, pero no menos importante, a todos los compañeros con los que he compartido tiempo a lo largo de la carrera: Aarón, Ester, Ryan, Carlos, María, Miriam, Raquel, Rubén, Álvaro, Víctor, Alberto, Paco…gracias por haberme acompañado en este viaje y por haber hecho de mi estancia en la universidad una etapa memorable.

A todos, gracias.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

ASDU *Application Service Data Units* (Unidades de Datos de Servicios de Aplicación), bloques de datos de la capa de aplicación de DNP3.

APDU *Application Protocol Data Units* (Unidades de Datos del Protocolo de Aplicación), ASDU que integran la cabecera de la capa de aplicación DNP3

DNP3 *Distributed Network Protocol Version 3* (Protocolo de Red Distribuida Versión 3), protocolo de comunicación para redes SCADA.

DSL *Domain Specific Language* (Lenguaje Específico del Dominio), lenguaje de programación destinado a resolver un problema concreto.

EPA *Enhanced Performance Architecture* (Arquitectura de Rendimiento Mejorado), Modelo de capas utilizado en DNP3.

HMI *Human Machine Interface* (Interfaz Humano Máquina), sistema a través del cual el operador humano controla los procesos de un sistema de monitorización.

HPCN *High Performance Computing and Networking Researching Group* (Grupo de Investigación de Redes y Computación de Altas Prestaciones)

IoT *Internet of Things* (Internet de las Cosas), Paradigma de comunicación en el que los objetos cotidianos se conectan a la red de Internet.

IED *Intelligent Electronic Device* (Dispositivo Electrónico Inteligente), Controlador basados en un microprocesador cuyo uso se enfoca a los sistemas de control

IDS *Intrusion Detection System* (Sistema de Detección de Intrusiones), Programa que monitoriza la red en busca de actividades maliciosas o intrusiones no autorizadas.

IP *Internet Protocol* (Protocolo de Internet), principal protocolo de comunicación de Internet.

LPDU *Link Protocol Data Units* (Unidades de Datos de Protocolo de Enlace), bloques de datos de la capa de enlace de DNP3.

MTU *Master Terminal Unit* (Unidad Terminal Maestra), host central que envía instrucciones al resto de elementos en una red SCADA.

NFC *Near Field Communication* (Comunicación de Campo Cercano), Tecnología de comunicación inalámbrica de corto alcance.

PLC *Programmable Logic Controller* (Controlador Lógico Programable) Computador digital usado para la automatización de procesos electromecánicos.

RFID *Radio Frequency Identification* (Identificador de Radio Frecuencia), Dispositivos de radiofrecuencia cuyo propósito fundamental es identificar un objeto físico.

RTU *Remote Terminal Unit (*Unidad Terminal Remota), dispositivo electrónico que puede obtener datos del entorno y enviarlos a otra unidad remota.

SCADA *Supervisory Control And Data Acquisition* (Supervisión, Control y Adquisición de Datos), Sistemas de comunicaciones que permiten controlar procesos a distancia.

TCP *Transmission Control Protocol* (Protocolo de Control de Transmisión), protocolo fundamental de Internet, perteneciente a la capa de transporte.

TPDU *Transport Protocol Data Units* (Unidades de Datos de Protocolo de Transporte), bloques de datos de la capa de transporte de DNP3.

1. Introducción

1.1 Motivación

La Internet de las Cosas surge a principios del milenio como una idea a través de la cual el modelo de Internet pasaría del estado actual, orientado a los clientes, a una conectividad total enfocada en las cosas, los objetos interconectados a la red.

Este modelo, poco valorado durante los primeros años tras su publicación, es hoy en día es una de las tendencias de investigación más importantes en el mundo y una de las líneas de negocio con mayor inversión en las empresas tecnológicas. La razón principal por la que la Internet de las Cosas ha ganado popularidad como modelo es el enorme crecimiento de los dispositivos conectados a Internet en los últimos diez años, hasta el punto en que ya hay más nodos en Internet que seres humanos sobre el planeta. [1]

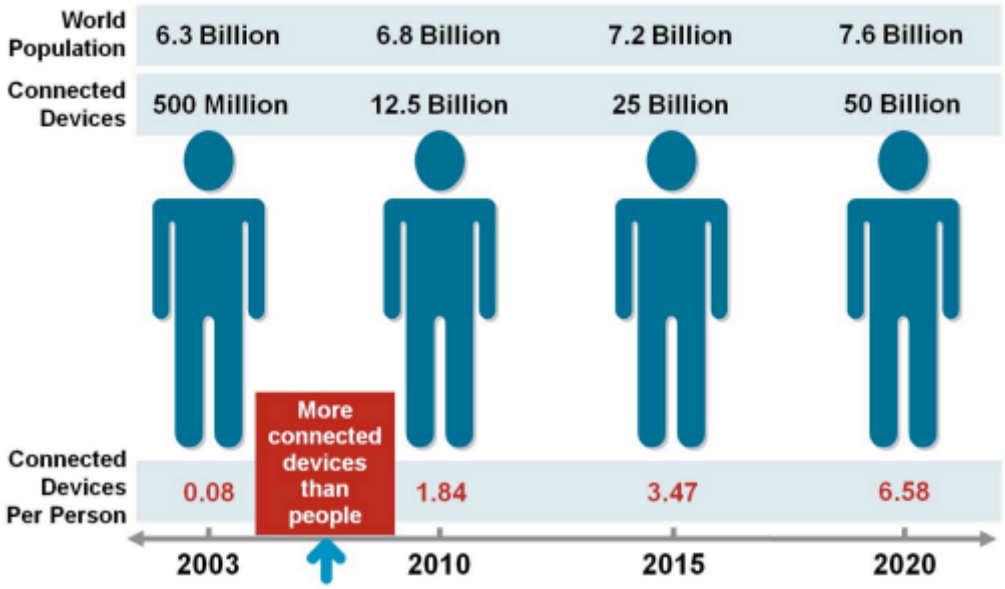

Figura 1 - Comparativa de crecimiento de la población humana y los dispositivos conectados a Internet [1]

Después de observar las cifras, resulta evidente por qué la Internet de las Cosas tiene tanto potencial como evolución de Internet. De ahora en adelante ya no serán las personas las encargadas de generar la mayor parte de datos de la red, sino que serán los propios objetos los que recogerán toda esa información del entorno, la compartirán y tendrán capacidad de actuación condicionada a los datos que reciban.

Las aplicaciones que pueden surgir de este nuevo paradigma son muy extensas; domótica, *smart grids*, automóviles inteligentes, nuevos medios para el *e-learning* y la telemedicina, sistemas de mantenimiento autónomos y otros avances hasta ahora impensables en nuestro día a día.

A pesar de las grandes ventajas que podría suponer la implementación global de IoT, debemos tener en cuenta las grandes implicaciones y riesgos de ciberseguridad que podrían surgir en un sistema de estas características. La necesidad de asegurar la protección de estos sistemas es capital, puesto que una red IoT insegura ampliaría

enormemente el alcance de los ataques informáticos, permitiendo por primera vez en la historia que los objetos físicos de la red puedan ser un objetivo. El requerimiento de un alto nivel de ciberseguridad puede justificarse también desde una perspectiva económica, debido a que los ataques informáticos tienen también un impacto monetario cada vez mayor en las empresas, las instituciones y los usuarios finales.

Los mecanismos de seguridad que se han utilizado hasta ahora en la red no son completamente efectivos para garantizar la protección de una red IoT. Es por eso que debemos investigar nuevas herramientas de seguridad que permitan un despliegue seguro que cumpla con las condiciones básicas de la seguridad informática; integridad, confidencialidad y disponibilidad.

La principal motivación de este Trabajo Fin de Grado es, por tanto, desarrollar un sistema de detección de tráfico anómalo para el protocolo DNP3 que sirva como complemento a otros mecanismos de seguridad tradicionales y permita revelar ataques e intrusiones contra la red, alertando al operador de la misma y en última instancia posibilitando que se pueda actuar en consecuencia.

1.2 Objetivos del Trabajo Fin de Grado

El objetivo de este Trabajo Fin de Grado es el diseño e implementación de un sistema de detección de tráfico anómalo para el protocolo DNP3 que pueda servir como complemento a otros sistemas de seguridad estándar para una aplicación IoT. Lo que queremos conseguir a través de esto es obtener un incremento en el nivel de seguridad de la implementación, de manera que el uso de un protocolo aparentemente inseguro como éste quede compensado.

Este mecanismo de monitorización tendrá que servir para reconocer ataques propios del protocolo DNP3, que no son detectables a priori por los mecanismos de patrón o firma que emplean los dispositivos de detección de intrusiones tradicionales. La particularidad de estos ataques, denominados comúnmente en la literatura como ataques complejos, es que se componen de una sucesión de comandos que por sí mismos no entrañan ningún peligro de seguridad, pero que al enviarse en un determinado orden son capaces de llevar a la máquina o al conjunto general del sistema a un estado crítico.

En aras de cumplir las motivaciones del Trabajo Fin de Grado, se han definido los siguientes objetivos:

- Investigar y recabar información sobre el protocolo DNP3, que faciliten su comprensión y permitan entender a un nivel avanzado cómo funciona el estándar.

- Diseñar y preparar un entorno de trabajo que permita la simulación virtual de una red SCADA de complejidad media operando con el protocolo DNP3, el cual se utilizará como soporte para desarrollar y probar el sistema de detección.

- Obtener conocimientos sobre el desarrollo de herramientas de detección de ataques y manipulación de paquetes en lenguaje Python, de tal manera que sea posible desarrollar los módulos necesarios para obtener la funcionalidad deseada en este proyecto.

- Desarrollar un sistema de monitorización que sirva como complemento a los IDS tradicionales. El sistema, como ya se ha mencionado, debe ser capaz de detectar ataques complejos sobre el protocolo DNP3 y mostrar alertas al operador en tiempo real. Cumplir este objetivo tendría como finalidad proveer de ayuda al operador central de una red IoT, permitiéndole detectar eventos sospechosos y actuar en consecuencia.

- Investigar y utilizar sistemas, técnicas y aplicaciones que permitan explotar vulnerabilidades de seguridad en estos protocolos. El uso posterior de todo este conocimiento será enfocado de cara a probar el funcionamiento del sistema.

- Plantear un escenario experimental donde pueda aplicarse el sistema previamente desarrollado a la seguridad de una red IoT. El cumplimiento de este objetivo permitirá validar su utilidad en una aplicación real y extraer conclusiones sobre su funcionamiento.

- Hacer una reflexión sobre el trabajo futuro en este ámbito, focalizando en el cambio del modelo actual de Internet a uno basado en los objetos y en la vital importancia de asegurar niveles altos de seguridad y confidencialidad para que este cambio sea posible.

- Escribir una memoria en la que quede constancia del proceso llevado a cabo y de los conocimientos adquiridos durante la realización del Trabajo Fin de Grado.

1.3 Fases de realización

Para el desarrollo y correcta consecución de este proyecto, se han seguido una serie de fases que se detallan a continuación:

- **Documentación**: En esta primera fase se realizó un estudio del concepto de IoT y su estado del arte, haciendo especial hincapié en artículos sobre su relación con los protocolos SCADA. También se hizo necesario investigar sobre el estado del arte y los entresijos de DNP3 con el fin de ser capaces de entender su modelo de comunicación. Por último, se llevó a cabo una exhaustiva búsqueda de información acerca de los distintos tipos de ataques que son propios de este protocolo y de cómo llevarlos a cabo, lo que nos permitió tratar de reproducirlos para estudiar el comportamiento del sistema ante ataques de estas características.

- **Preparación del entorno de trabajo**: Durante la fase de preparación se estudiaron las diferentes alternativas de software disponibles para crear una red DNP3 simulada en la que basar nuestras pruebas. Tras comparar las distintas características de cada una de las herramientas, se eligió una de ellas y se creó el entorno de trabajo mediante el uso de máquinas virtuales.

- **Diseño e implementación del *sniffer*:** En la etapa de diseño e implementación se desarrollaron los programas y bibliotecas que constituirían el sistema final de detección de tráfico anómalo para el protocolo DNP3.. Durante este ciclo se avanzó en la creación de los diferentes módulos que integran el sistema de detección, partiendo de un *sniffer* básico en lenguaje Python cuyas

funcionalidades de detección fueron mejoradas iterativamente hasta obtener el nivel de desarrollo esperado.

- **Realización de pruebas:** Durante la fase de realización de pruebas se comprobó que la funcionalidad desarrollada en el sistema de detección de tráfico anómalo funcionaba de manera correcta y acorde a los objetivos planteados en el tercer capítulo de este Trabajo Fin de Grado. Esta fase incluye también la corrección de errores de implementación que se detectaron durante los tests.

- **Verificación de resultados:** En esta etapa se ideó y desarrolló el escenario de aplicación para el sistema de detección de tráfico anómalo. En este marco de ámbito realista se probaron los distintos ataques para los que se habían diseñado mecanismos de monitorización y detección de anomalías, permitiéndonos extrapolar datos de efectividad para otras aplicaciones. A partir de los resultados obtenidos se enuncian las nuevas ideas, la problemática que queda por resolver y el trabajo futuro.

- **Redacción de la memoria:** Por último, se redactó este documento para dejar plasmada la experiencia y el conocimiento adquiridos durante la realización del Trabajo Fin de Grado.

A continuación se muestra un diagrama de Gantt que posibilita un mejor seguimiento de los tiempos empleados en el cada uno de los periodos del proyecto:

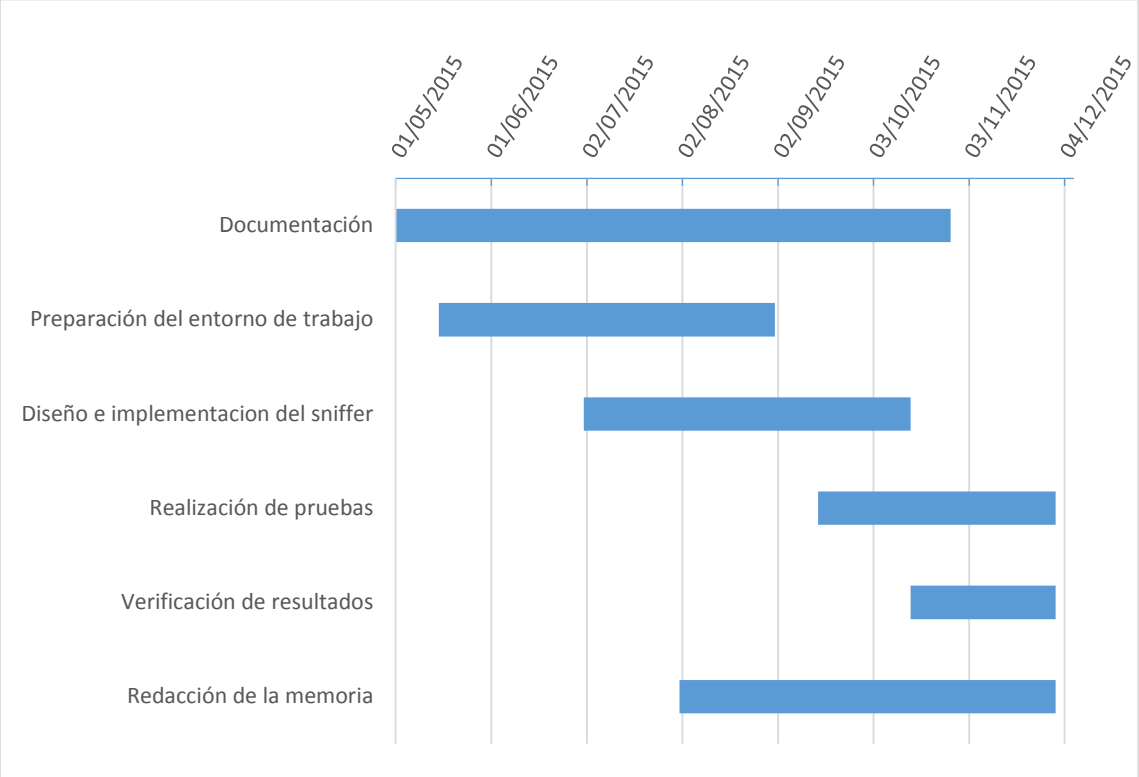

Figura 2 - Diagrama de Gantt relativo al proyecto

1.4 Estructura del documento

A continuación se detalla la estructura de este Trabajo Fin de Grado:

1) En el capítulo 2 se presentará una revisión de todas las tecnologías implicadas en el estudio, desarrollo e implementación del sistema con el fin de comprender mejor el contexto del trabajo. Se profundizará especialmente en el concepto de IoT, en el de redes SCADA y en las relaciones entre ambos. También será especialmente importante recabar información sobre el protocolo SCADA específico en el que se centra el trabajo, DNP3. El cierre de este apartado irá dedicado a una revisión del estado del arte de los sistemas de detección de intrusiones o IDS.

2) Una vez explicadas las generalidades mediante el estado del arte, se pasará a analizar el problema que motiva el TFG en el capítulo 3, haciendo hincapié en las características técnicas y en los requisitos, tanto funcionales como no funcionales, que se han tenido que satisfacer para llevar el desarrollo a buen puerto.

3) Después de haber analizado a fondo los requisitos y la motivación del TFG, se hablará de la solución propuesta y de cómo ésta ha sido desarrollada para solventar la motivación del trabajo en el capítulo 4. En este capítulo se detallarán los pasos realizados en la creación del sistema de detección de tráfico anómalo propuesto siguiendo un enfoque incremental, partiendo de las generalidades y explicando una a una todas las características del programa.

4) En el capítulo 5 se detallará el proceso de validación del sistema, especificando los diferentes casos de prueba, los resultados esperados y los finalmente obtenidos para tener una idea de la efectividad de la solución propuesta.

5) Para finalizar el documento, en el capítulo 6 se plantearán las conclusiones finales del proyecto y del trabajo futuro que a partir de éste pudiera realizarse.

2. Estado del arte

2.1 Introducción

En este capítulo se expone el actual estado del arte en materia del Internet de las Cosas, las redes SCADA, el protocolo DNP3 y los sistemas de detección de intrusiones, todos ellos directamente relacionados con la implementación del sistema de monitorización de tráfico objeto de este Trabajo Fin de Grado.

El objetivo del capítulo es dar al lector una visión más amplia del contexto vigente en estas tecnologías, definiendo el marco de conocimiento necesario para la realización del proyecto.

2.2 Internet de las Cosas

El término *Internet of Things*, Internet de las Cosas en castellano, hace referencia a varios aspectos acerca de la extensión de la red y de Internet al mundo de los objetos físicos, posible gracias a la aparición generalizada de dispositivos con capacidad sensórica y de actuación. [2]

El concepto de IoT, introducido a finales de la década de los noventa por Kevin Ashton, es multidisciplinar y se ve afectado por distintos puntos de vista [3]. Esto ha suscitado controversias dentro de la comunidad científica y ha implicado que la idea asociada a IoT haya ido modificándose durante los últimos quince años, aunque a día de hoy la idea de *"un paradigma, varias visiones"* parece ser la más aceptada.

De acuerdo a esta idea, IoT puede ser definido mediante la intersección de tres perspectivas diferentes; "orientada a los objetos", "orientada a internet" y "orientada a semántica" como muestra la siguiente figura:

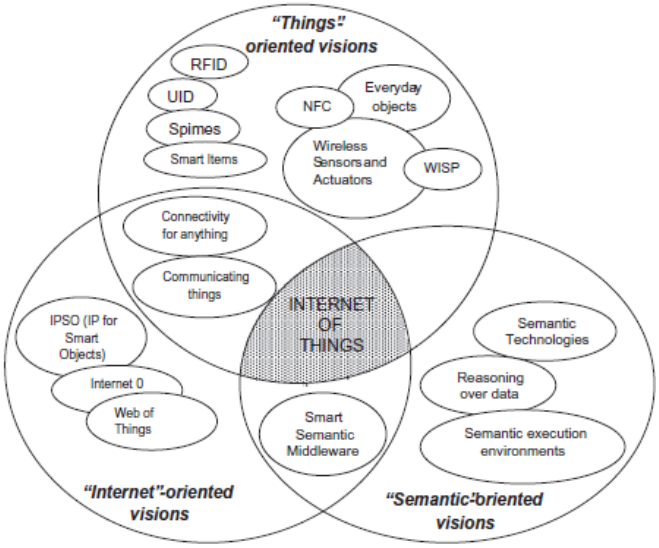

Figura 3 - Intersección de las distintas visiones de IoT [2]

La visión orientada a los objetos se centra en el nivel de componente unitario, valga la redundancia, el objeto. Dichos objetos comparten una serie de características, entre las que podemos encontrar: [4]

- Son tangibles, es decir, forman parte del conjunto de objetos físicos reales.
- Disponen de, como mínimo, las capacidades de comunicación relativas a recibir y contestar mensajes. También poseen mecanismos que permiten que otros dispositivos los localicen y contacten.
- Poseen un identificador único.
- Tienen mecanismos de computación, lo que les permite procesar información de diversa índole en función de la capacidad del objeto.
- Poseen medios para detectar magnitudes físicas. Algunos podrán ser configurados para comportarse también como actuadores que desencadenen una acción en el mundo real.

La visión orientada a Internet focaliza en los conceptos de la red IoT, caracterizándola como un conjunto muy grande de nodos en un sistema distribuido. Los nodos recopilan información del mundo físico y la traducen a streams digitales para su envío, de modo que será de esperar un alto volumen de peticiones y respuestas de datos entre los diferentes puntos de la red. Teniendo en cuenta el volumen de información, será importante proveer a la red de medios que garanticen su escalabilidad y estabilidad.

La intersección entre la visión orientada a Internet y la visión orientada a objetos parte de un objetivo claro, unir de alguna manera los objetos físicos con las topologías de red para lograr que exista comunicación entre todas las cosas que nos rodean. Para que esto sea posible necesitaremos no sólo una red capaz de asignar direcciones únicas a cada uno de los objetos, sino dotar a los objetos cotidianos de mecanismos de conectividad.

Si pensamos por un momento en la cantidad de objetos cotidianos a los que podríamos dotar de conectividad para incluirlos en el paradigma IoT, podemos prever que deberemos mantener un grupo muy grande y heterogéneo de nodos enviando, recibiendo y almacenando información. Por ende, necesitaremos mecanismos propios de la visión semántica para buscar, almacenar, conectar y representar toda la información generada por la red. Dentro de la rama semántica se tratan también los temas de estandarización y definición de lenguajes para la comunicación en IoT.

2.2.1 Computación ubicua e Internet de las Cosas

La visión de IoT se alinea perfectamente con el paradigma de computación ubicua, donde la informática se integra en el entorno de la persona eliminando la frontera entre los ordenadores y el resto de objetos para crear entornos inteligentes. A pesar de que este concepto está sujeto a cierta discusión, la definición más aceptada de entorno inteligente es "*el mundo físico que se entreteje invisiblemente con sensores, actuadores, displays y elementos computacionales integrados a la perfección en los objetos cotidianos de nuestras vidas y conectados a través de la red*", la cual fue propuesta por Mark Weiser en 1999. [5]

La idea de ubicuidad en la computación no es reciente, ya que las primeras referencias a este modelo datan de finales de los años ochenta. Hoy en día la popularización de la

tecnología RFID y la evolución de la electrónica digital permiten el diseño y la fabricación de dispositivos en miniatura capaces de adquirir información y comunicarse entre sí, proporcionando los pilares necesarios para el despliegue de redes IoT. Desde un punto de vista taxonómico, una red IoT bajo el paradigma de computación ubicua estará compuesta por *Cosas*, objetos que actuarán como los diferentes nodos de la red. Dichos objetos pueden abstraerse en tres niveles diferenciados: [6]

I. **Nivel hardware:** Engloba los sensores, actuadores y hardware empotrado de comunicación. En el ámbito IoT, el componente hardware más reseñable es la etiqueta RFID, la cual ha supuesto un tremendo salto hacia delante en el ámbito de la comunicación inalámbrica, estando presentes a día de hoy en sistemas de reparto, en aplicaciones de control de accesos e incluso en tarjetas inteligentes. Las etiquetas RFID se clasifican como pasivas si no disponen de mecanismos de alimentación propios y utilizan la energía de la señal del lector para poder comunicarse, o como activas, si disponen de fuente de alimentación integrada, siendo las pasivas las más frecuentes por su menor coste de implementación.

II. **Nivel middleware:** Se centra en el almacenamiento bajo demanda y el análisis de datos, ambos fundamentales dados los altísimos volúmenes de información generados en estos modelos de red. Los datos obtenidos deben ser procesados de manera óptima mediante técnicas de inteligencia artificial -como las redes neuronales- que posibiliten el monitoreo de datos sin verse saturados por las altas tasas de tráfico. Dada la heterogeneidad de los redes, será fundamental también garantizar que estos algoritmos pueden trabajar de forma centralizada o distribuida, según requiera la aplicación.

III. **Nivel de presentación:** Contiene las herramientas de visualización y representación que permiten el acceso a los datos de las distintas plataformas interconectadas. Es en este nivel donde sucede la interacción entre el usuario final y el entorno, por lo que engloba también mecanismos de detección de eventos y de extracción de información útil a partir de datos en crudo del sensor.

2.2.2 Seguridad en IoT

La seguridad se convierte en un punto extremadamente relevante en la Internet de las Cosas, al mismo nivel incluso que el desarrollo de los medios que permitan integrar dispositivos de comunicación de bajo tamaño y coste a los objetos, o del despliegue de redes que sean capaces de manejar tráfico de fuentes de tipo diverso. En un mundo que es víctima constante de ataques informáticos y con una conciencia cada vez mayor de la importancia de la ciberseguridad, IoT nunca despegará si no garantiza un alto nivel de protección a sus usuarios.

La dificultad para aplicar técnicas de ciberseguridad a IoT radica en la cantidad y la variedad de elementos presentes dentro de un ecosistema, lo que dificulta la aplicación de mecanismos clásicos de defensa y aumenta mucho el área de ataque. Prevenir vulnerabilidades será ahora un problema multidisciplinar que englobará áreas muy diferentes; seguridad de protocolos y redes, privacidad de datos, gestión dinámica de confianza, tolerancia a fallos y gestión de identidades entre otras. [7]

El mejor medio para garantizar la protección de una red IoT es asegurar que todos los niveles que la forman cumplan una serie de mínimos de seguridad. En este proyecto nos centramos en la seguridad relativa a los protocolos de red utilizados en las comunicaciones, pero es importante tener en cuenta la vital importancia de bastionar otras áreas tales como la integridad de los datos almacenados, la privacidad de la comunicación y la gestión de las identidades.

2.3 SCADA

Dentro de las implementaciones de IoT, uno de los sistemas más desplegados es SCADA. Los sistemas SCADA se utilizan para controlar y monitorizar infraestructuras críticas de toda índole basándose en el uso de protocolos de comunicación, sistemas de sensores y mecanismos de computación, cumpliendo así las características de las redes IoT mencionadas en la sección anterior. Su uso permite la intercomunicación y el control de todos los elementos que componen el sistema, posibilitando la compartición de datos y el desencadenamiento de acciones si es necesario. Este tipo de protocolos se ha venido desarrollando desde los años 60 en aplicaciones típicas del ámbito industrial, siendo especialmente importante en los sistemas de distribución de electricidad y agua, en la industria energética y en los sistemas de telecomunicaciones [8]. Por este motivo, dentro del desarrollo del sistema de monitorización que es objetivo de este proyecto, nos centraremos en este tipo de protocolos.

El catálogo de protocolos SCADA disponibles para una implementación de estas características ofrece muchas alternativas, tales como DNP3, Modbus, ICCP u OPC. Todos estos protocolos son muy diferentes entre sí y están orientados a distintas aplicaciones. Sin embargo, tienen una serie de características en común: [9]

- Control de acceso: Los usuarios se encuentran dentro de grupos con privilegios de lectura y escritura específicos sobre los parámetros del proceso y las funcionalidades del producto.
- Manejo de alarmas: Los sistemas SCADA disponen de alarmas cuyo manejo se basa en la comprobación del estado en los diferentes nodos y en la comparación con un valor límite. Las alarmas se gestionan de manera centralizada, de tal forma que la información que manejan solo existe en un lugar y todos los usuarios ven el mismo estado. La mayoría de implementaciones permiten establecer distintos niveles de alarma y generar un correo electrónico en respuesta a la información procesada en un evento.
- Archivado de registros o *logs*: Una de las funcionalidades básicas de estos protocolos es el archivado de *logs*. Durante el tiempo de ejecución de un programa SCADA se van guardando datos relevantes de la comunicación acompañados de una marca de tiempo y un identificador que permite su filtrado y estudio posterior.
- Generación de informes: La mayoría de las aplicaciones SCADA disponen de herramientas de reporte compatibles con hojas de cálculo o sistemas de bases de datos.
- Automatización: Una de las capacidades más potentes de SCADA es la de ejecutar acciones automáticamente cuando se produzca un evento determinado. Estos desencadenantes se programan mediante lenguajes de scripting propios de cada protocolo y pueden dar lugar a secuencias de acción muy complejas en uno o varios dispositivos del sistema.

2.3.1 Arquitectura

Los sistemas SCADA tienen una arquitectura variable en función del protocolo utilizado y de la aplicación para la que han sido diseñados, pero todos pueden englobarse dentro del modelo que puede verse de manera esquemática en la figura siguiente. La función de cada una de las partes se pormenoriza en las siguientes subsecciones.

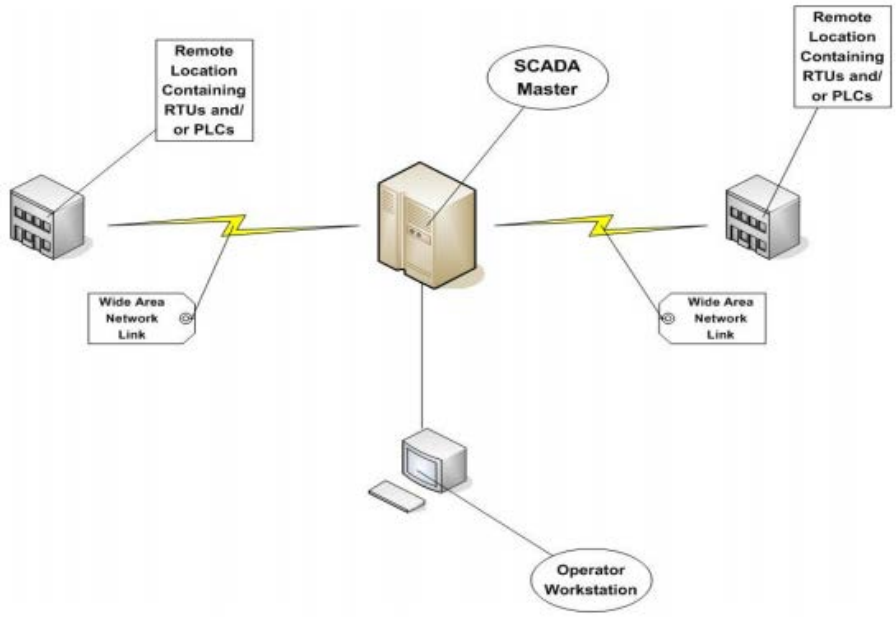

Figura 4 –Vista esquemática de la arquitectura SCADA [8]

2.3.1.1 - Unidades Terminales Remotas (*RTU*)

Las RTU son los dispositivos que, repartidos a lo largo de toda la infraestructura de red, se ocupan de adquirir información útil para la monitorización y el control de la misma. Estas terminales, que en el argot tradicional de las redes se considerarían como *esclavos*, adquieren información de su entorno y la traducen al lenguaje o protocolo usado en la comunicación SCADA para su envío a un centro de procesamiento o *maestro*.

Podemos diferenciar los RTU en función de la aplicación que desarrollan, siendo los más usuales los siguientes tipos:

- Módulos analógicos de entrada/salida: Se ocupan de la detección y la actuación en el ámbito de las variables continuas. Los módulos de entrada tienen un número de interfaces cuyo valor típico suele ser 4, 8, 16 o 32. Los módulos de salida toman valores digitales de la CPU y lo convierten a representaciones analógicas para su envío a los actuadores.

- Módulos digitales de entrada/salida: Estos módulos se utilizan en el display de estados y la gestión de señales de alarma.

A modo de ejemplo, un RTU colocado en un transformador eléctrico de un sistema de distribución de energía podría tomar valores de voltajes o corrientes instantáneas, empaquetarlo en un mensaje de un protocolo de red y enviarlo al maestro para su procesamiento.

2.3.1.2 - Controladores Lógicos Programables (PLC)

Los PLC son dispositivos que se utilizan para automatizar y controlar procesos electromecánicos, desencadenando acciones físicas como mover un pistón o abrir una válvula. También se consideran dispositivos *esclavos*. Estos aparatos, surgidos como sustitutos del relé, se han venido utilizando en la industria durante los últimos cuarenta años y están diseñados para trabajar en condiciones térmicas, electromagnéticas y mecánicas desfavorables. A día de hoy se han ganado el papel de estándar hardware en aplicaciones de control.

Aunque para respetar los tecnicismos en esta revisión del estado del arte se ha preferido separar en dos categorías distintas a los PLC de los RTU, lo cierto es que en los últimos años las características técnicas de ambos dispositivos se han ido solapando y a día de hoy es difícil establecer una frontera clara entre los dos.

2.3.1.3 - Unidades Terminales Maestras (MTU)

El MTU hace la función de servidor central de todo el sistema, recibiendo datos de los RTU, extrayendo y almacenando la información relevante para la aplicación y procesando los datos de forma que sean entendibles para un operador humano. Desde el MTU se gestionan también las órdenes que los PLC deberán ejecutar en el mundo físico, siguiendo la política de eventos que el sistema tenga configurada.

La comunicación entre MTU y RTU es bidireccional, lo que quiere decir que el RTU podrá en la mayoría de los casos iniciar una comunicación con un MTU sin que éste lo haya hecho previamente. A pesar de esto, la gran mayoría del tráfico en las redes SCADA se basa en peticiones de datos –lecturas– que el MTU solicita a los RTU, por lo que se suele denominar como *maestro* a las *Master Terminal Units*. Dependiendo del protocolo SCADA utilizado en el sistema, puede haber uno o varios maestros, lo que puede derivar en distintas topologías de red.

2.3.1.4 - Red de comunicaciones

La red de comunicaciones de un sistema SCADA se encarga de asegurar la transmisión de los distintos mensajes y datos que se generan entre los elementos que conforman la red, como los RTU o los PLC. Si bien es cierto que la mayoría de implementaciones industriales utilizan un soporte físico como el cable por tener una mayor robustez frente a fenómenos electromagnéticos adversos, también pueden establecerse otros tipos de comunicación, como enlace inalámbrico o incluso satelital.

Desde sus orígenes las redes SCADA han ido evolucionando desde un modelo de comunicación típicamente pensado para puerto serie a uno IP, debido en gran parte a la influencia de los conceptos de la IoT. Esta evolución dota a los sistemas SCADA de mayor flexibilidad, pero a su vez implica la existencia de nuevas oportunidades disponibles para los hackers y es sin duda uno de los motivos que ha elevado el número de ciberataques sobre estos sistemas.

La elección del medio de transmisión más adecuado es crítica para diseñar una red con buen rendimiento y dependerá de la aplicación, del área de despliegue y en última instancia, del presupuesto.

2.3.1.5 – Interfaz Hombre Máquina

El operador humano necesitará tener acceso en todo momento a información relativa al sistema que está administrando. Para ello, es preciso que disponga de una máquina con conexión remota al MTU y que ésta sea capaz de mostrarle los datos en un formato amigable.

Para ello se emplea un sistema de Interfaz Hombre Máquina (*HMI*, por sus siglas en inglés), que muestra al operador la información en tiempo real y generalmente de forma gráfica. Las funciones más importantes de los sistemas HMI se listan a continuación: [10]

- Hacer accesible información técnica de alta complejidad a personal que disponga de pocos conocimientos técnicos, simplificando el tratamiento de la información.
- Monitorizar la disponibilidad del sistema y el tiempo online.
- Mantener un histórico de los parámetros de la red.
- Ayudar a los procesos de depuración y corrección de errores, aportando información útil.
- Permitir el análisis del sistema para determinar potenciales acciones que mejoren su rendimiento.

2.4 DNP3

2.4.1 Generalidades

DNP es un protocolo SCADA de ámbito tradicionalmente industrial creado por la empresa canadiense Westronic Inc. (actualmente GE Harris) a partir de la hasta entonces incompleta especificación del IEC 60870-5. Su tercera versión, DNP3, fue abierta como protocolo al público en 1993 y desde entonces su implantación y su desarrollo ha sido fundamental en infraestructuras críticas tales como la energía, el transporte, la gestión de residuos, el suministro de agua y otros servicios públicos. [11]

En los últimos veinte años se ha consolidado como uno de los estándares de facto en este tipo de aplicaciones bajo la tutela del *DNP Users Group*, que proporciona a sus miembros formación, soporte y documentación sobre el protocolo desde su sitio web.

DNP3 rivaliza en popularidad y cuota de mercado con otro protocolo que surgió del IEC 60870, el IEC 60870-5-101. Aunque IEC 60870-5-101 es el estándar más extendido en Europa, DNP3 es un protocolo más global, con presencia en Norteamérica, Sudamérica, Sudáfrica, Asia y Australia.

La mayor diseminación a nivel global del protocolo DNP3 es debida, entre otros factores, a que es un estándar de aplicación general en infraestructuras críticas, mientras que IEC 60870-5-101 está especialmente focalizado en el sector de la distribución eléctrica. En su amplia distribución influye también el hecho de que sea un protocolo abierto, lo que ha provocado que haya sido implementado en los sistemas hardware de múltiples fabricantes y que garantice, hasta cierto punto, la interoperabilidad entre ellos. Dicha interoperabilidad unida a la generalización del uso de este protocolo en la mayor parte del mundo suponen la justificación del gran número de implementaciones de la Internet de las Cosas que se plantean sobre DNP3.

El estándar DNP3 define el proceso de comunicación entre Unidades Terminales Maestras (MTU), Unidades Terminales Remotas (RTU) y otros *IEDs* (Dispositivos Electrónicos Inteligentes) en aplicaciones SCADA. La comunicación en este tipo de sistemas está orientada a la recolección de datos y al envío de comandos que desencadenan una acción para dar una respuesta física a la situación que describen los datos recibidos.

DNP3, a pesar de estar orientado a la transmisión de información, no está pensado para el envío de grandes volúmenes de datos sino para el envío fiable de pequeños paquetes asegurando en todo momento que éstos se reciben en un orden determinado. Podría decirse que el estándar al completo está diseñado para proporcionar una gran robustez al sistema de comunicaciones. Esto tiene sentido si se observa que su ámbito de aplicación principal son las infraestructuras críticas, donde las condiciones de transmisión no siempre son favorables y un fallo en la transmisión de un dato de una RTU o un comando de un master puede tener consecuencias muy graves

A continuación se enuncian las características principales de DNP3:

- Soporta mensajes con *timestamp* para mantener un sistema de secuencia de eventos.
- Permite implementaciones con múltiples maestros, así como el envío de mensajes no solicitados hacia el MTU.
- Define el tipo de datos que se manejan entre MTU y RTU a través de un modelo de objetos, que puede estar orientado a objetos y a aplicación. Este modelo es flexible y permite añadir variaciones en el formato, como por ejemplo la especificación del tamaño del valor.
- Tiene su propio modelo de capas, *Enhanced Performance Architecture (EPA)*, el cual es una simplificación del modelo OSI. Esta característica es muy importante y se profundizará en ella en la subsección siguiente.
- Soporta carga y descarga de archivos, lo que es útil para la gestión de archivos de configuración.
- Dispone de código de redundancia cíclica para el control de errores.

2.4.2 Modelo de capas

DNP3 utiliza un modelo de capas más simple que el de OSI (*Open Systems Interconnection*) para mejorar el rendimiento del sistema, llamado *Enhanced Performance Architecture (EPA)*. Mientras que el modelo de OSI se compone de las clásicas siete capas, el modelo DNP3 se reduce solamente a tres, donde las dos inferiores (capa de enlace y capa física) son hardware y solamente la de aplicación es software. [12]

En este modelo, al igual que en el OSI de siete capas, los datos de más alto nivel se sitúan en la capa de aplicación. En dicha capa se genera y se transmite toda la información para los usuarios, entre cuyos datos figuran eventos, salidas digitales, salidas analógicas, señales de alarma y estados del dispositivo que se transmiten entre los MTUs y RTUs que se encuentren dentro de la red. Es también en este nivel donde se sitúan los elementos que permiten a la información llegar de manera comprensible al operador humano de la red a través de software de Interfaz Hombre Máquina.

Inmediatamente por debajo se encuentra la capa de pseudo-transporte, que permite transmitir bloques de datos con un peso superior al permitido en una trama. Esta fragmentación utiliza funciones que estarían situadas dentro de la capa de red, como el routing, y funciones de la capa de transporte, como el control de flujo, por lo que el convenio estipulado es situarla en un plano que comprende a estas dos capas del modelo OSI.

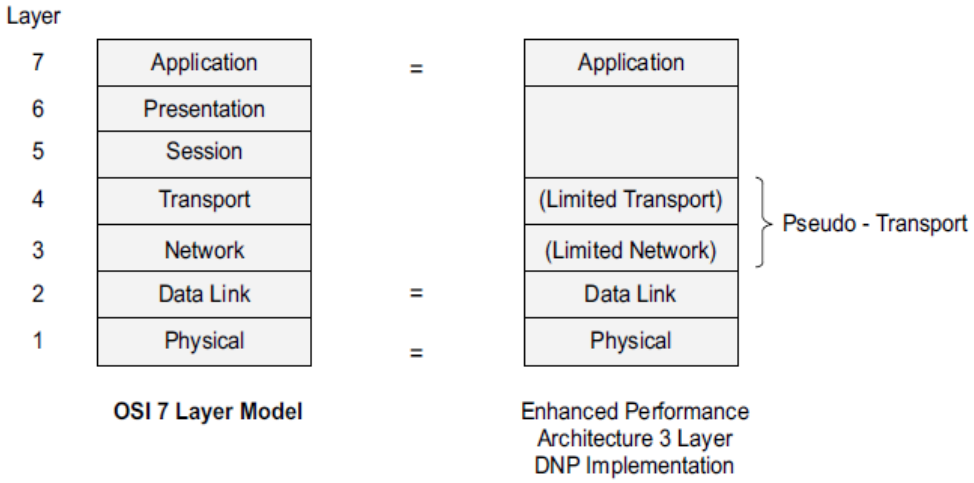

Figura 5 - Comparación entre el modelo de capas OSI y el usado en DNP3 [12]

Los datos ya fragmentados en la capa de pseudo-transporte se procesan en la capa de enlace, que proporciona medios para la transmisión fiable de datos a través de la red, como el control de flujos y la detección de errores.

La capa física es equivalente en su funcionamiento a la del modelo OSI, y debido a ello aunque está presente en la arquitectura *EPA* la literatura se refiere a ella como un modelo de tres capas.

2.4.3 Construcción de los paquetes

Tal y como ya se ha hablado previamente en la sección 2.3.2, DNP3 utiliza un modelo *EPA* de tres capas para dar formato a los mensajes. Como es usual en los protocolos de red, cada capa del modelo toma la información de la capa superior y le añade una cabecera, aumentando el tamaño del paquete. En recepción el proceso es equivalente pero de sentido inverso, el mensaje se procesará capa a capa eliminando las cabeceras y extrayendo la información que sea relevante en cada una de ellas.

En la capa de aplicación se procesan los mensajes de más alto nivel. Estos mensajes se dividen en bloques de datos de tamaño manejable denominados *Application Service Data Units* (ASDU), a partir de los cuales se crean los *Application Protocol Data Unit* (APDU) mediante la adición de una cabecera. Dicha cabecera puede ser de 2 o 4 bytes, en función de si el mensaje es una petición o una respuesta. En el caso de una petición se envía solamente la cabecera sin ASDU.

El tamaño de los APDU está limitado a 2048 bytes, por lo que si el tamaño del mensaje es superior se hará necesario recurrir a las capacidades de fragmentación de la capa de pseudo-transporte. En dicha capa, cada APDU del nivel superior se dividirá en varias

unidades menores del protocolo de transporte llamadas *Transport Protocol Data Units* (TPDU), las cuales constan de un byte de cabecera seguido de, generalmente, 249 bytes de datos.

En el nivel de capa de enlace se toman los TPDU y se les añade una cabecera de 10 bytes y un código de redundancia cíclica de 16 bits para aportar el mecanismo de detección de errores. A partir de estas modificaciones, el TPDU se convierte en la unidad básica de la capa de enlace, denominada *Link Protocol Data Unit* (LPDU). El tamaño de los LPDU está fijado a 292 bytes por razones históricas de compatibilidad con otros protocolos SCADA, como el IEC-870-5. La siguiente figura pretende ser un esquema aclaratorio del proceso de construcción de un mensaje DNP3 a través de sus distintas capas.

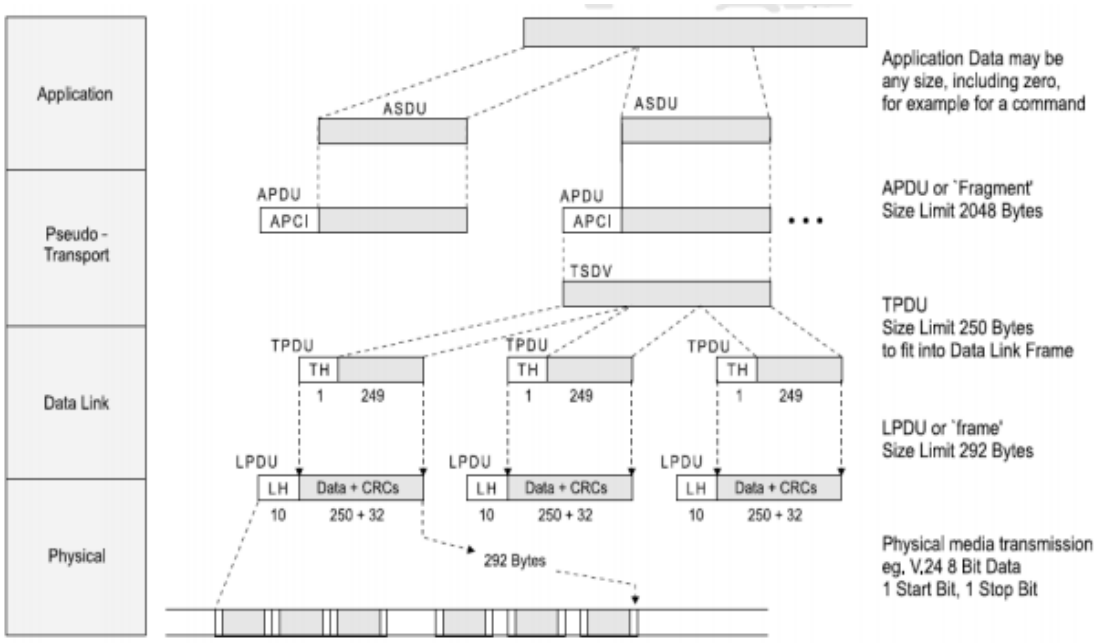

Figura 6 - Construcción de un mensaje DNP3 [13]

2.4.4 Seguridad

La seguridad en los protocolos SCADA, incuestionable hasta principios de siglo, se ha convertido recientemente en objeto de debate y discusión en el ámbito investigador y profesional. El experimento Aurora, llevado a cabo por el *Idaho National Laboratory* en 2007, provocó la destrucción física de un generador eléctrico a partir de un ciberataque simple y constituyó el detonante de una incipiente alarma social en el ámbito de la ciberseguridad en infraestructuras críticas. [14]

A medida que el problema de seguridad en estos protocolos ha ido ganando relevancia se han ido descubriendo vulnerabilidades que pueden afectar a estos sistemas, la mayoría de las cuales son fácilmente explotables y además suponen un gran impacto y alcance que se extiende al mundo de lo físico.

DNP3, como protocolo SCADA elegido para el estudio en este proyecto, adolece de carencias en seguridad desde el momento de su concepción como estándar. Dado que es un protocolo con más de veinte años, se priorizó en su diseño la fiabilidad de la comunicación por encima de la seguridad, que no suponía un problema grave en aquel entonces.

El protocolo DNP3 presenta tres deficiencias fundamentales:

1- No posee mecanismos para comprobar la integridad de los paquetes. Es decir, que el paquete no ha sido modificado intencionadamente durante su tránsito por la red.
2- No dispone de mecanismos de autenticación entre maestro y esclavo, lo que abre la posibilidad que alguien usurpe la identidad del maestro.
3- No posee mecanismos para evitar la repetición de comandos.

Las primeras aplicaciones sobre DNP3 confiaban la seguridad de la implementación al aislamiento de la red, ya que las conexiones se realizaban sobre puerto serie. A partir del momento en que el estándar se actualiza para funcionar sobre TCP/IP debido a las necesidades de IoT, las implementaciones DNP3 van quedando cada vez más y más expuestas a acciones externas y se convierten en vulnerables a las tres deficiencias mencionadas.

Suponiendo que el atacante tiene acceso al medio de transmisión, podemos clasificar los ataques en cuatro tipos bien diferenciados. Los cuatro tipos, que pueden observarse de manera esquemática en la figura 7, se detallan a continuación:

- Interrupción: El mensaje de la fuente es bloqueado por un atacante situado entre origen y destino, evitando que alcance al receptor.

- Intercepción: El mensaje de la fuente llega al destino, pero también al atacante situado entre origen y destino. Los ataques de intercepción suelen desarrollarse como paso previo a otro ataque, con el fin de obtener información sobre la red, los nodos que la componen y la naturaleza de la información.

- Modificación: El mensaje de la fuente es interceptado y modificado por un atacante para luego ser reenviado hacia su destino original. Al no existir mecanismos de autenticación, el receptor del mensaje no tiene manera de comprobar que el emisor es quien dice ser.

- Fabricación: El atacante crea un mensaje y lo envía directamente al receptor haciéndose pasar por un emisor legítimo.

En vista de los tipos de ataque, uno puede pensar que la aplicación de sistemas de seguridad tradicionales supondría la solución a la mayoría de estas vulnerabilidades, pero no es así. DNP3, y sus homólogos en SCADA, se ven afectados por los llamados ataques complejos, que tienen la característica de estar constituidos por una secuencia de paquetes no maliciosos a nivel individual, pero que en conjunto son capaces de llevar al sistema a un estado crítico. Es por eso que los sistemas clásicos basados en reconocimiento de patrones no son suficientes para detectar todos los ataques posibles en este protocolo. Algunos investigadores sugieren que la manera más efectiva para poder detectar este tipo de ataques es usar detectores semánticos, que mantienen un registro del estado actual del sistema y son capaces de determinar si un paquete a priori válido tendrá como consecuencia la entrada en un estado inconsistente. [15]

Este enfoque es preciso pero requiere de gran esfuerzo computacional además de la necesidad de conocer a priori la estructura de la red, lo que lo hace poco práctico para

aplicaciones IoT donde la integración del sistema de seguridad tiene que ser eficiente y barata. En este proyecto, por tanto, nos basamos en la estacionariedad del protocolo DNP3 para tratar de detectar ataques a partir de fluctuaciones en el tráfico de unas características determinadas.

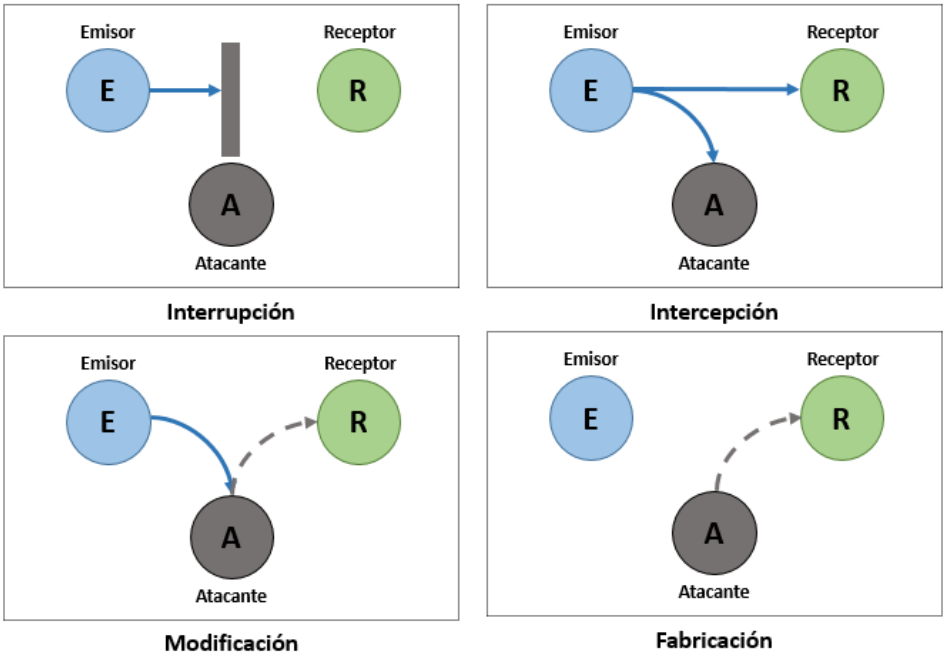

Figura 7 - Tipos de ataque en DNP3 [18]

Para terminar la revisión del estado del arte de la seguridad de DNP3, se va a mencionar el *fuzzing*, técnica que consiste en enviar una serie de paquetes en los que unos campos determinados varían de forma aleatoria o semi-aleatoria, buscando provocar una reacción inesperada en el sistema receptor. Este tipo de ataque, que entraría dentro de la categoría de fabricación, se suele utilizar por los desarrolladores para la detección de errores, por lo que es fácil encontrarlos en Internet y se considera que su uso está al alcance de cualquier usuario con ciertas capacidades informáticas. [23]

2.5 Sistemas de Detección de Intrusiones

2.5.1 Introducción

Un IDS es un sistema que permite determinar cuándo se están produciendo intentos sin autorización de penetrar en una red y detectar ciertos tipos de ataques sobre la misma. Es importante matizar que, por sí mismo, el IDS no se encarga de detener los ataques, sino de notificar o bien al operador o bien a otro elemento de la red que se encargue de eliminar la amenaza. En caso de detener ataques, se estaría hablando de un IPS (*Intrusion Protection System*).

Mientras que en los comienzos de la era Internet, el uso de un firewall perimetral en una red era suficiente para considerar que ésta estaba correctamente defendida frente a amenazas externas, hoy en día el aumento de las funcionalidades que ha experimentado la red global dificulta enormemente el delimitar con claridad los perímetros y por tanto lo que el cortafuegos debe o no bloquear. Debido a esto, los IDS han venido ganando popularidad en los últimos años, a medida que también ha ido aumentando la conciencia

general sobre la importancia de la seguridad informática. A día de hoy, estos sistemas son los más utilizados para detectar ataques de todo tipo y podemos encontrar una gran variedad de ellos, algunos incluso bajo licencias de software libre (*Snort, Bro.*…).

En el contexto de este Trabajo Fin de Grado, desarrollaremos un sistema de detección y monitorización de tráfico que pueda complementar la funcionalidad de un IDS tradicional adaptándolo al protocolo SCADA que ya se ha tratado en puntos anteriores, DNP3, para su aplicación en una implementación de la IoT.

2.5.2 Funcionamiento

A pesar de que, como ya se ha comentado, existen múltiples variantes y desarrollos de los sistemas IDS con sus respectivas particularidades, en esta sección se va a tratar de ofrecer al lector una visión panorámica de su funcionamiento.

En función de donde esté colocado el IDS, la literatura suele referirse a él como HIDS (*Host-based)* o NIDS *(Network-based)*. Los primeros suelen ser sistemas que incluyen herramientas antivirus convencionales, firewall y algunos sistemas de prevención de intrusiones. Para las aplicaciones IoT en las que se centra este proyecto, nos centraremos en los segundos, NIDS, aunque nos referiremos a ellos en todo momento como IDS por simplicidad.

Los mecanismos de detección de intrusiones escanean tráfico de red de manera promiscua en una interfaz, buscando actividad sospechosa basándose en firmas o patrones de los paquetes analizados. Para determinar si un paquete es o no sospechoso, un IDS dispone de mecanismos que comparan la firma del paquete con una lista de firmas de ataques. Estas listas, disponibles en Internet, disponen ya de miles de registros de ataques y crecen día tras día, por lo que será fundamental para el buen funcionamiento del sistema de detección que sean periódicamente actualizadas. [16]

Uno de los objetivos más importantes de los IDS es la detección precoz de los paquetes malintencionados, ya se esté produciendo una intrusión o un ataque. A pesar de que muchas veces la detección en tiempo real no es completamente realista y se produce cierto retraso, el que la alarma del sistema se levante a tiempo podrá ayudar al operador del sistema a prevenirlo antes de que cause daños graves.

Los IDS también proveen de otra herramienta fundamental para el administrador de la red, los archivos de log. Estos archivos, que guardan información útil acerca del tráfico que se ha producido durante el tiempo en que el IDS ha estado activado, pueden ayudar a prevenir un ataque que haya sucedido interpretando la secuencia de eventos.

2.5.3 Configuración

La configuración del IDS es una pieza fundamental para el correcto funcionamiento del sistema y el valor de los parámetros que la definen, que dependerán del entorno en el que se vaya a aplicar, deberán estudiarse antes de su implementación final.

Cuando un algoritmo detecta cualquier tipo de actividad de red normal como ataque, se considera que se ha cometido un falso positivo. En el otro extremo, si un algoritmo de detección no detecta un ataque como tal, diremos que se ha producido un falso negativo.

Ambos fenómenos son no deseados y deben reducirse al mínimo mediante el ajuste de los valores de umbral en las comprobaciones.

A pesar de que puede parecer que los falsos negativos son más peligrosos que los falsos positivos, es necesario entender que si el sistema genera alertas de manera muy frecuente por tener un umbral muy bajo, probablemente el operador acabe haciendo caso omiso de las advertencias. Debido a este hecho, se hace necesario configurar los valores de umbral de los sistemas de detección de intrusiones para que respondan de manera contundente, pero no excesiva, a los distintos eventos detectados.

3. Análisis del problema

3.1 Introducción

En este tercer capítulo se aborda el estudio del problema relativo al TFG profundizando en el establecimiento de los requisitos que debemos cumplir para superarla. El objetivo de esta sección es dar al lector una visión más completa de lo que se pretende conseguir al finalizar su desarrollo.

3.2 Análisis de requisitos

En esta subsección se listan los requisitos propuestos para el sistema que se va a desarrollar, diferenciando los funcionales de los no funcionales. Esta separación tiene como intención contextualizar lo máximo posible cada requisito para que su alcance quede correctamente delimitado. La tabla de clasificación de los requisitos puede encontrarse en el Anexo A.

3.2.1 Requisitos funcionales

El sistema de monitorización y detección de tráfico anómalo para tráfico DNP3 sobre el que versa este trabajo tiene una serie de requisitos funcionales que lo rigen y que definen las acciones que se espera que pueda llevar a cabo. Estos requisitos pueden organizarse en una serie de módulos independientes, tal y como puede verse a continuación:

- Captura del tráfico: El sistema debe contar con una función de monitorización de tráfico DNP3 que permita analizar el flujo de mensajes paquete a paquete y extraer valores que sean de interés para la construcción de métricas. Será importante que el sistema *sniffer* no sufra pérdidas de paquetes elevadas y que la velocidad de procesamiento sea lo suficientemente alta como para tratar sin problemas todos los mensajes generados en un escenario IoT de complejidad media.

- Mecanismo de alarma: Debe proveerse al sistema de un mecanismo que sea capaz de analizar los datos obtenidos de la monitorización de los flujos de mensajes y determinar, a través de patrones y cálculos estadísticos, si la situación de tráfico es normal o se está produciendo algún tipo de ataque. Además, será condición que esto pueda realizarse en tiempo real, alertando al operador justo en el momento en que se produzca el ataque.

- Display de los datos en tiempo real: El programa mostrará por pantalla alertas en tiempo real cuando se detecten trazas de un ataque o intrusión y cuando la situación del estado general de alarma en el sistema sufra un cambio. En cada uno de estos eventos, se mostrarán parámetros adicionales como el instante temporal, el grado de severidad y un identificador del evento.

- Resumen final: En el momento en que se reciba un comando de parada por parte del operador del sistema, deberán mostrarse como resumen los datos más representativos de lo acaecido durante el proceso de monitorización.

- Utilidad de configuración: Para el ajuste de los parámetros del sistema de monitorización y detección de tráfico anómalo, se proveerá de un archivo de configuración en formato *.ini* que permitirá al usuario modificar de manera sencilla los valores por defecto para adecuar el sistema a una aplicación concreta.

3.2.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no son relativos a comportamientos específicos del sistema, sino que hacen referencia a aspectos generales del mismo. A continuación se detallan los que se han de cumplir durante el desarrollo del Trabajo Fin de Grado.

Entre los requisitos no funcionales más importantes se encuentran la fiabilidad y la estabilidad, de tal manera que el sistema sea capaz de obtener datos precisos y procesar alertas relativas a la detección de eventos de forma fiable y prolongada en el tiempo, con requisitos de funcionamiento cercanos al tiempo real. El rendimiento forma también parte importante de los requisitos no funcionales, ya que el sistema desarrollado debe disponer de capacidad para procesar los mensajes de manera eficiente y ser capaz de soportar tasas de comunicación medias sobre el protocolo DNP3. El sistema debe orientarse en todo momento a la sencillez y la inmediatez de la recogida de datos, ya que la información que se procese debe ser entregada de manera rápida, efectiva y sencilla a un operador sin excesivas capacidades informáticas. El código desarrollado deberá estar correctamente documentado y regirse por los principios de alta cohesión y bajo acoplamiento para que el sistema pueda ser mejorado en un futuro por otro programador. Para terminar, se hace hincapié en la capital importancia que reviste el que el sistema pueda implementarse con un coste económico bajo en una red IoT para que el despliegue de la misma siga siendo viable en términos económicos.

3.3 Conclusiones

En esta sección han quedado definidos los distintos requisitos que habrán de cumplirse durante el desarrollo del proyecto, diferenciando entre los funcionales y los no funcionales. Estos requisitos servirán como guía para la correcta finalización del mismo y se estudiarán para evaluar si se han cumplido o no los objetivos que se buscaban con la realización de este Trabajo Fin de Grado.

El procedimiento a seguir para el desarrollo de la aplicación en base a dichos requisitos funcionales y no funcionales se expone en profundidad en el próximo capítulo.

4. Desarrollo de la solución

4.1 Introducción

En este capítulo se detallará el proceso de creación del sistema de detección de tráfico anómalo para que ésta cumpla los requisitos expuestos en el capítulo anterior. Para una mayor coherencia y facilidad de lectura, se dividirá la funcionalidad en varias partes:

- Generalidades
- Sistema de captura de tráfico
- Análisis de parámetros de los paquetes
- Mecanismos de alarma
- Visualización de resultados

Con el fin de ajustar el contenido del Trabajo Fin de Grado al máximo posible a la normativa, la información relativa al entorno de trabajo utilizado para el desarrollo de la solución se encuentra disponible en el Anexo B.

4.2 Generalidades

El sistema de detección y monitorización de tráfico anómalo en DNP3 se compone de una serie de scripts íntegramente escritos en lenguaje Python en su versión 2.7. El hecho de haber elegido esta versión responde a razones de robustez con las dependencias de red que mantiene, pero no debería suponer un problema de compatibilidad con otras versiones superiores. El uso del programa está enfocado a entornos Linux.

Dicho script Python se valdrá de la herramienta *Scapy*, cuya funcionalidad se describe en detalle en el Anexo C. *Scapy* es una herramienta de manipulación interactiva de paquetes que permite el uso de funciones tales como crear tráfico capa a capa, enviar y recibir flujos de paquetes y capturar el tráfico en una interfaz mediante mecanismos de sniffing.

Tanto *Scapy* como los módulos desarrollados tendrán como prerrequisito de ejecución la instalación de las bibliotecas *libpcap* y *libdnet* en su versión más reciente para Python. Estas bibliotecas, en las que se basan gran parte de las herramientas de red que tienen que ver con la captura de paquetes, sustentan el funcionamiento del conjunto general del programa.

El programa se ejecutará invocando el script *sniff.py* desde línea de comandos con permisos de súper-usuario, lo que permitirá al programa tener acceso a la captura de paquetes sobre la interfaz *eth0*. La funcionalidad completa del programa en sus distintos módulos se detalla en los puntos siguientes, siendo especialmente importantes la captura del tráfico, el análisis de parámetros del tráfico, la gestión de alarmas y la visualización de los resultados.

4.3 Sistema de captura del tráfico

Al lanzarse el programa se inicializan las variables y las ventanas de captura de paquetes que se utilizarán para almacenar los datos resultantes de la monitorización. De cara a cumplir los objetivos en materia de adaptabilidad del programa, es en este momento donde se produce la primera llamada al archivo de configuración *config.ini*, que dota al usuario de una gran flexibilidad en el uso del programa sin tener que recurrir a modificaciones internas del código.

El proceso de obtención de los datos de configuración se realiza a través del módulo de Python *ConfigParser*, que proporciona un método de recogida de variables en archivos con una estructura similar a los archivos de extensión *ini* de Microsoft Windows®. Antes de iniciar la ejecución, el usuario deberá ajustar los parámetros de configuración a los valores que sean más convenientes para su aplicación en concreto. Entre otras cosas, son parametrizables el tamaño de la ventana de observación, el protocolo de capa de transporte predeterminado, el puerto a utilizar en las comunicaciones y el factor de peso de cada uno de los ataques que se pueden detectar.

El siguiente paso del programa es importar la biblioteca *DNP3_Lib.py*, que aumenta la funcionalidad de *Scapy* permitiéndole efectuar operaciones sobre paquetes DNP3, capacidad no incluida en la utilidad por defecto. Esta biblioteca, creada en el contexto de la investigación sobre seguridad en infraestructuras críticas, implementa un modelo simplificado de las distintas capas de DNP3 para *Scapy*, haciendo posible al programa diseccionar los paquetes de este protocolo y pudiendo extraer directamente de ellos los valores incluidos en las cabeceras de las distintas capas. [17]

Una vez establecidos todos los parámetros de configuración inicial e importada la ampliación de *Scapy* para DNP3, se enlaza todo el tráfico dirigido o proveniente del puerto *dnp3_port* a la clase DNP3 de *Scapy*, permitiendo al programa tratar la comunicación DNP3 como tal. Como ya se ha mencionado, el puerto *dnp3_port* es parametrizable a través del archivo de configuración, aunque por defecto se utiliza el 20000.

En este punto, la configuración inicial estará terminada y podrá empezar el proceso de captura mediante la función *sniff()*, que en esta aplicación concreta tiene como objetivo la interfaz *eth0* de la máquina donde se ejecuta el programa. Al comenzar a funcionar el sniffer se guardan también valores de reloj que serán útiles para generar estadísticas sobre tiempos de ejecución al finalizar el proceso.

La función *sniff()* captura todos los paquetes de la interfaz de uno en uno, pasándolos consecutivamente como argumento a otra función que se encarga de realizar sobre ellos ciertas comprobaciones y extraer valores generales. Estas operaciones se verán con detalle en la siguiente sección.

4.4 Análisis de parámetros de los paquetes

Como se ha visto en el punto anterior, mediante el *sniffer* implementado cada paquete entrante es procesado de forma individual para extraer una serie de parámetros que permitirán al sistema decidir si el tráfico es o no legítimo y manejar las alarmas. Podemos definir dos tipos de parámetros, globales y específicos.

4.4.1 Parámetros globales

Los parámetros globales que se muestrean y analizan se pueden ver en la siguiente lista:

- Número total de paquetes.
- Número de paquetes IP.
- Número de paquetes TCP.
- Número de paquetes UDP.
- Número de paquetes DNP3.

El objetivo del conteo de este tipo de parámetros es poder generar un análisis de los ratios de cada tipo de paquete frente al total, lo que será de utilidad para poder determinar si el tráfico en un momento dado está siendo normal o se está produciendo alguna alteración. Estas métricas se utilizan también para generar el histórico de datos del programa con los resultados de la captura del tráfico que se muestran cuando el operador decide finalizar la ejecución del programa.

4.4.2 Parámetros específicos

A diferencia de los anteriores, que se usan en términos más estadísticos, los parámetros específicos que se extraen hacen referencia a campos muy concretos de los paquetes que se ven afectados en determinados ataques, permitiendo detectarlos por mecanismos de reconocimiento de firma. Estos parámetros se utilizarán para detectar ataques comunes del protocolo DNP3 que se han hecho de dominio público a partir de distintas investigaciones realizadas en los últimos años. [18]

En la tabla que se muestra a continuación se listan los parámetros específicos que se obtienen durante la ejecución del programa, relacionándolos con el ataque que se pretende detectar a través de su estudio:

Tabla 1 - Listado de parámetros DNP3 cuya detección es objetivo

Parámetro	Ataque relacionado
Dirección IP de origen	Generalmente en este tipo de sistemas se suele asignar direcciones IP consecutivas a los elementos de la red. Si la distancia entre la dirección de origen y de destino es superior a un umbral predefinido un número elevado de veces, es posible que el sistema esté sufriendo un ataque.
Flag DFC en la capa de enlace	El flag DFC de la capa de enlace sirve para indicar que el esclavo está ocupado y no puede atender la petición. Puede utilizarse por un atacante para inutilizar un RTU.
Función *Reset Link States*	La función Reset Link States se utiliza para identificar si un punto de la red está operativo. Puede ser utilizado por un agresor para conocer el estado de un potencial objetivo de ataque.
Dirección IP de destino	Si la dirección IP de destino es 255.255.255.255 se produce un mensaje broadcast a todos los dispositivos de la subred. Puede utilizarse por un atacante para saturar la red.
Código de función *write*	Si se producen sucesivas escrituras en un RTU, puede darse lugar a *overflow* en los *buffers* y a errores inesperados, dependiendo del tipo de implementación del sistema.

Código de función *clear objects*	Los mensajes con código de función 9 (*clear objects*) y 10 (*clear objects sin asentimiento*) en la capa de aplicación borran todos los datos del RTU. Es especialmente peligroso en su variante sin asentimiento, ya que puede llevar con facilidad al sistema a un estado inconsistente.
Código de función *reset*	Los mensajes con código de función 13 (*cold restart*) y 14 (*warm restart*) en capa de aplicación provocan el reinicio de los esclavos. Un atacante puede aprovechar estos códigos para mantener el sistema en un estado continuo de reinicio o, aún peor, provocar la reinicialización de los RTU en un estado inconsistente.
Código de función *initialize data*	Los mensajes con código de función 15 en capa de aplicación provocan la reinicialización de los objetos de las RTU. Un atacante podría utilizar mensajes fabricados con este código de función para corromper los datos de un terminal.
Código de función *outstation application termination*	Los mensajes con código de función 18 en capa de aplicación causan la terminación de los procesos que la RTU esté ejecutando. Un atacante podría valerse de este tipo de mensajes para provocar que los terminales aparezcan como bloqueados y no respondan a las órdenes del operador legítimo.
Código de función *delete file*	Si un mensaje DNP3 tiene código de función 27, la RTU que lo reciba borrará los archivos que esté manejando. Esta función, utilizada por un atacante, puede provocar pérdidas de información, bloqueos del sistema y además puede facilitar la eliminación de huellas del ataque si se consigue eliminar los *logs* de tráfico que se almacenan en los RTU.
Quinto bit del campo *IIN*	Si el quinto bit del campo IIN en capa de aplicación está activo, se notifica al MTU de que el archivo de configuración del RTU está corrupto para que el MTU reenvíe una copia. Un usuario malicioso podría valerse de esto para interceptar una copia del archivo de configuración, obteniendo valiosa información sobre el sistema o reenviando después una copia modificada del archivo de configuración al RTU.

Como puede verse en la lista anterior, los parámetros obtenidos hacen referencia a ciertos ataques comunes en el protocolo DNP3. En la siguiente sección se detallará el proceso que siguen los mecanismos de alarma del programa para identificar estos ataques y ponderar la relevancia que puedan tener en el desempeño del sistema.

4.5 Mecanismos de alarma

Los mecanismos de alarma del sistema de detección están definidos en el módulo *AlarmSystem*, al que se invoca cada vez que hay que hacer una estimación sobre la evidencia de un ataque y su implicación en el sistema.

La idea de estos mecanismos de alarma se basa en el mantenimiento de estados de riesgo para la operación del conjunto mediante una máquina simple de cuatro estados. Cuando se alcanza determinado nivel de riesgo global, la máquina de estados eleva el estado de alerta, y si por el contrario el tráfico se normaliza, lo reduce.

Esta máquina de estados se divide en los cuatro niveles de alerta que se definen a continuación:

1- Nivel de alerta bajo: Este estado representa el nivel de amenaza más bajo posible. Puede implicar tráfico completamente normal o tráfico normal con algunas anomalías de escasa importancia que no afectan al normal funcionamiento del sistema y no suponen un riesgo grave de seguridad. La situación de ejecución más deseable sería que el sistema de detección se mantuviera siempre en este estado.

2- Nivel de alerta medio: El nivel de alerta medio supone el segundo nivel de riesgo del sistema de detección. Encontrarnos en este estado implica que el sistema de detección está hallando parámetros de tráfico no permitidos con una frecuencia que puede suponer cierto riesgo de seguridad. Llegado a este nivel, el operador debe actuar para poner una solución al problema que se esté detectando, aunque los niveles de riesgo aún no sean críticos.

3- Nivel de alerta alto: El estado alto es el tercer nivel de riesgo en la máquina de estados, y la entrada en él supone la confirmación de que se está produciendo una vulneración de la seguridad del sistema. En este punto el operador de la red puede descartar prácticamente la posibilidad de que se esté produciendo un falso positivo, y debe dar alta prioridad a la búsqueda de soluciones.

4- Nivel de alerta crítico: El estado más alto en la jerarquía de riesgos empleada en el sistema corresponde con el crítico. Representa una peligrosidad todavía mayor que el estado de alerta alto e implica un ataque severo contra el sistema, siendo posible incluso que se esté produciendo una combinación de varios ataques simultáneos. El operador debe actuar en consecuencia de manera inmediata, planteándose incluso la desconexión por mantenimiento.

La detección de los ataques se produce a través de un sistema de ponderación de amenazas que se basa en una ventana de *N* valores por cada ataque que se pretende detectar donde *N* es un número entero que se define en el archivo de configuración previamente mencionado. Con cada paquete analizado, la ventana de un determinado ataque almacena un '1' si se detectan los parámetros típicos de ese ataque y un '0' si no se detectan.

Una vez transcurrido cierto tiempo de ejecución que asegure que la ventana, que actúa a efectos prácticos como una cola FIFO, se ha llenado se calcula el factor de amenaza que implica este ataque. A partir de los datos de la ventana se obtiene un factor de frecuencia *F* que nos da la información de la cantidad de repeticiones registradas de dicho ataque en el tiempo y un factor *R* que aporta información acerca de la cantidad de repeticiones del ataque ocurridas en los últimos instantes de ejecución. El estado general de la amenaza del sistema relativo a cada ataque viene dado por la siguiente ecuación, donde *P* es el factor de peso elegido por el usuario en el archivo de configuración:

$$Amenaza = F * R^2 * P$$

Como se puede deducir, según este planteamiento el estado de alerta que se calcula para cada ataque viene determinado en su mayor parte por las detecciones más recientes debido a que el término R se encuentra elevado al cuadrado, aunque también se tiene en cuenta el histórico de ocurrencias y el factor de peso. El estado general del sistema viene

determinado por la suma de los factores de amenaza de cada ataque que se han calculado de manera independiente. Este valor numérico sirve de base a la máquina de estados del sistema de alarma para situar el nivel de amenaza.

Además de la gestión global del estado de alarma a través de la máquina de estados ya mencionada, el sistema dispone de un mecanismo de notificación individual de alertas. Esta herramienta, típicamente incluida en los sistemas IDS, permite avisar al operador de que ha ocurrido un evento anómalo y complementa al ya mencionado sistema global de alertas, aportando parámetros específicos del ataque detectado como la fecha del suceso, la gravedad y un identificador del suceso detectado.

Queda patente en este método la idea de emplear valores parametrizables por el usuario a través del archivo de configuración, ya que este añadido proporciona al programa una gran capacidad de acomodación a distintos escenarios y permite al administrador focalizar en determinados ataques y pasar por alto otros.

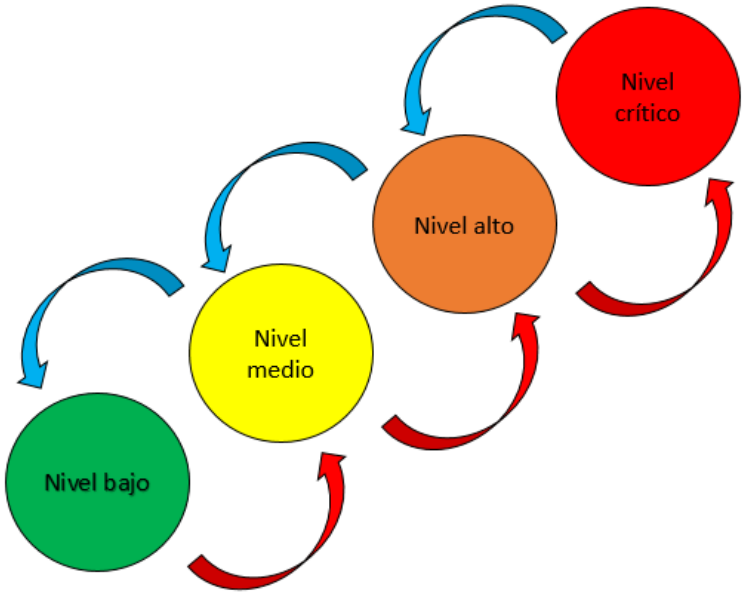

Figura 8 - Modelo de la máquina de estados empleada en el mecanismo de alarma

En el siguiente apartado se aclarará como el programa gestiona la salida por pantalla de las distintas alarmas y datos obtenidos durante la ejecución.

4.6 Visualización de resultados

El módulo de visualización de resultados se encarga de la tarea de transmitir todos los datos resultantes de la detección de tráfico al operador para que éste pueda actuar en consecuencia. El diseño tiene como objetivo proporcionar salida en tipo texto plano de dos maneras distintas, según convenga:

- Salida por pantalla de terminal: Tiene objetivo de proveer de información en tiempo real a la persona encargada de analizarla. Los mensajes que se mostrarán serán relativos a anomalías detectadas durante la ejecución del programa y también sobre cambios en el estado del nivel de alarma global.

- Volcado a archivo *log* de texto: De acuerdo con los objetivos propuestos en la sección 3, la salida por pantalla del programa se producirá en formato TSV para poder ser analizada a posteriori de manera sencilla con algún programa de cálculo. El objetivo de esta funcionalidad es permitir a los gestores de la red el mantenimiento de un histórico o la comparación de una traza en un momento dado con otra guardada con anterioridad.

En relación a la salida por pantalla de terminal, el criterio de diseño ha sido el de tratar de reducir al máximo el número de mensajes que se muestran por pantalla sin que eso repercuta en un ocultamiento de información. El motivo de esto es evitar que la persona encargada de responder a las alarmas se vea saturada por un exceso de información y pase por alto las notificaciones del sistema de alarma.

En el momento en que el operador pone fin a la ejecución del programa se genera también un pequeño resumen que proporciona datos estadísticos de la ejecución, tales como el tiempo de captura, el número de paquetes totales capturados, el porcentaje de tráfico DNP3 y el tráfico de fondo. Este resumen se realiza a partir de los parámetros globales pormenorizados en el apartado anterior.

4.7 Conclusiones

Mediante los módulos detallados en los puntos anteriores se ha tratado de conseguir la funcionalidad marcada por los objetivos de este Trabajo Fin de Grado. Al concluir este apartado se da por finalizada la parte de desarrollo e implementación, disponiendo de un disector eficaz para el protocolo DNP3 que trabaja conjuntamente con el gestor de alarmas y la visualización de datos ya analizados en detalle.

De cara a probar la efectividad del programa desarrollado, se examina en la siguiente sección un escenario realista de aplicación del sistema en una red IoT. A través del estudio del comportamiento del sistema en estas condiciones, podremos obtener las capacidades reales del sistema y proponer modificaciones que puedan aplicarse como trabajo futuro para mejorar su rendimiento.

5 Validación

5.1 Introducción

En este capítulo del Trabajo Fin de Grado se analizará la efectividad de la solución desarrollada en un escenario de aplicación real, con el fin de medir el grado de cumplimiento de los objetivos funcionales y no funcionales además de la capacidad técnica del sistema para detectar amenazas.

Además de la propia validación del proyecto realizado, este estudio será fundamental para analizar las alternativas y mejoras que puedan implementarse como trabajo futuro y para extraer las conclusiones finales del proyecto.

En los siguientes sub-apartados se contextualizará el escenario de aplicación para situar al lector de manera precisa en el ambiente de las pruebas y se detallarán en profundidad las pruebas realizadas y los resultados obtenidos.

5.2 Escenario de validación

Como escenario de validación se propone una vivienda con capacidades domóticas. La casa en cuestión dispone de múltiples objetos inteligentes conectados entre sí, formando una red que opera bajo el paradigma de la Internet de las Cosas. Uno de los objetos más importantes de esta red será el contador eléctrico inteligente, que sustituye al contador electromecánico tradicional y permite la recolección de datos de manera telemática, la monitorización de valores como la potencia instantánea o el amperaje y la gestión de alarmas en caso de avería. [19]

El modelo de comunicación empleado en Smart Grids requiere garantías de interoperabilidad, por lo que se emplean en él protocolos SCADA tales como DNP3, IEC 61850 u OPCUA [20]. De cara a la validación del sistema diseñado se asumirá un Smart Meter con capacidad de comunicación DNP3 sobre TCP/IP. El enlace con la central se producirá por cable cruzado RJ45.

Dado que el contador es el encargado de gestionar la entrada de energía eléctrica al domicilio y de reportar los consumos a la empresa proveedora de electricidad, es un punto estratégico en la logística del hogar y debe tenerse en cuenta como uno de los posibles objetivos de un ciberataque. Un atacante que consiguiera acceso a la red podría utilizar instancias comunes de ataques del protocolo DNP3 para eliminar registros de la memoria, impedir el envío de datos a la central, enviar valores falsos, levantar alarmas de avería inexistentes e incluso ocultar problemas técnicos reales, por lo que este elemento se convierte en un punto crítico de la seguridad del hogar.

La comunicación legítima entre la central y el Smart Meter se efectúa mediante la herramienta de simulación de protocolos industriales *Axon Test*, cuya funcionalidad completa queda descrita en el Anexo C, a través de dos instancias de este programa:

- Un emisor DNP3 representando la central eléctrica, que efectúa una serie de peticiones periódicas de datos al Smart Meter.

- Un receptor DNP3 representando al Smart Meter, con capacidad para contestar las peticiones de datos de la central y enviar mensajes no solicitados, tales como alarmas, indicadores de avería, etc.

Para la realización de las pruebas de validación hemos desarrollado un programa en lenguaje Python, denominado *DNP3Crafter*, con capacidad para efectuar ataques sobre el protocolo DNP3 a partir del envío de paquetes fabricados con características específicas. Este script hace uso de los sockets de Scapy y permite elegir el ataque deseado de una lista y el número de repeticiones del mismo.

DNP3Crafter está diseñado para lanzar ataques complejos en DNP3, los cuáles consisten en el envío de paquetes genéricos que no representan una amenaza de manera individual pero sin embargo sí lo hacen como conjunto o sucesión de dichos paquetes. La carga útil y el CRC de los paquetes generados están precalculados y no varían, por lo que su uso no representa un test de *fuzzing*. El establecimiento de conexión, el incremento del número de secuencia de los paquetes y el resto de problemática asociada a la conexión TCP se resuelve de manera sencilla mediante el uso del socket, que automatiza todos estos procesos.

```
david@david-VirtualBox ~/Escritorio/Emisor Scapy $ ./DNP3Crafter.py 192.168.56.101
 ___   __    ___  _____   ___                 __ _
   /   \ /\ \ \ / _ \|___ /   / __\ __  __ _  / _|| |_  ___  _ __
  / /\ //  \/ // /_)/  |_ \  / /  | '__|/ _` || |_ | __|/ _ \| '__|
 / /_// /\  // ___/  ___) |/ /___| |  | (_| ||  _|| |_|  __/| |
/___,' \_\ \/ \/     |____/ \____/|_|   \__,_||_|   \__|\___||_|

Choose one action to perform:
1: Health check
2: Warm Restart attack
3: Cold Restart attack
4: Write attack
5: Initialize data attack
6: App function termination attack
7: Delete file attack

5
Choose number of repetitions:
3

Sent 1 repetitions...
Sent 2 repetitions...
Sent 3 repetitions...
Finished.
```

Figura 9 - Ejemplo de ejecución de ataque DNP3 en DNP3Crafter

En el siguiente apartado se efectuarán distintos ataques arquetípicos de este protocolo sobre un enlace de comunicaciones simulado entre el contador inteligente y la central eléctrica, teniendo en cuenta que el contador tendrá implementado el sistema de detección desarrollado a lo largo del proyecto.

5.3 Pruebas realizadas

Las pruebas se clasifican en dos secciones distintas; pruebas generales y ataques complejos. En la primera se comprueba el funcionamiento en sentido amplio del sistema de detección, incluyendo la detección del tráfico de fondo y los parámetros relacionados con la dirección IP de los paquetes DNP3, así como el rendimiento del sniffer en términos de paquetes procesados. En la relativa a los ataques complejos se comprueba la respuesta del sistema ante un evento de ataque DNP3 específico.

5.3.1 Pruebas generales

La primera prueba realizada sobre el sistema de detección ha consistido en la medida de la capacidad del sniffer para procesar tráfico. Como Scapy no dispone de una herramienta de conteo de los paquetes perdidos en el kernel de forma nativa, para esta prueba se ha monitorizado una comunicación entre emisor y receptor que mediante Wireshark y el sistema desarrollado. Después de tres minutos de interacción entre ambas partes, el número de paquetes detectados en ambos programas es equivalente y se confirma que los paquetes DNP3 están siendo interpretados de manera correcta en el sistema de detección. Estos resultados eran de esperar dado que el escenario plantea una tasa de transferencia de mensajes moderada.

La segunda prueba es relativa al rendimiento de la aplicación en el procesamiento de paquetes a distintas tasas de envío. Para comprobar esto se enviaron flujos de quinientas escrituras DNP3 a través de la aplicación *DNP3Crafter*, reduciendo paulatinamente los espacios temporales entre el envío de cada paquete en el emisor. Los resultados obtenidos nos muestran que la aplicación es capaz de procesar altos flujos de paquetes sin tener pérdidas hasta que el tiempo entre paquetes es de 35 ms, lo que supone un rendimiento aceptable solo en tasas medias y bajas. El bajo rendimiento en esta prueba es justificable dadas las limitaciones en velocidad del lenguaje Python en comparación con otros lenguajes de programación. El listado completo de resultados de la prueba puede comprobarse en la tabla correspondiente del Anexo D.

La tercera prueba consiste en la detección de situaciones en las que la distancia entre la dirección IP de emisor y receptor exceda un rango cuyo valor es configurable. Para la realización de esta prueba se ha otorgado a un emisor DNP3 legítimo una IP que sobrepasa el valor establecido en el archivo de configuración y se ha lanzado la comunicación, previendo obtener una serie de alertas por pantalla. Como se esperaba, los mensajes que cumplen esta anomalía son detectados por el sistema y levantan alertas de nivel medio.

La cuarta prueba se basa en la detección del tráfico de fondo a la comunicación legítima. En algunos casos de aplicación puede ser útil mantener un registro de los tipos de paquetes detectados en la red de cara a un estudio estadístico posterior, por lo que el sistema de detección desarrollado ha sido dotado de un mecanismo de conteo y almacenamiento de estos valores y su display a modo de resumen. Para comprobar el correcto desempeño de esta función, se simula una comunicación análoga a la de la primera prueba para analizar el display de resultados del programa la cual se para cinco minutos más tarde. Al parar la ejecución del programa, el módulo de display de resultados muestra los valores de tiempo empleado y del total de paquetes procesados, separando en distintos módulos los paquetes IP, TCP, UDP y DNP3 del resto. En el anexo C pueden observarse varios ejemplos de la recopilación e impresión de resultados, al final de cada captura.

5.3.2 Ataques complejos

En esta sección se prueba la respuesta del sistema de detección que hemos desarrollado ante seis ataques complejos típicos del protocolo DNP3 lanzados a través del programa *DNP3Crafter*. El funcionamiento de cada ataque, la metodología empleada para llevarlo a cabo y los resultados obtenidos se detallan a continuación. En el Anexo D se puede ver un volcado completo de la salida por pantalla del sistema de detección en cada uno de los casos.

1- Warm Restart Attack

El ataque *Restart Attack* en modalidad *warm* consiste en el envío periódico de mensajes con código de función 0x0E en capa de aplicación. Cuando estos mensajes llegan al RTU se efectúa un reinicio parcial de la secuencia de comunicaciones, lo cual posibilita el uso de un flujo de mensajes con esta función para efectuar un ataque de denegación de servicio.

Para comprobar la reacción del sistema de detección a este tipo de ataques, se ha generado un flujo de treinta paquetes de estas características hacia el receptor. Los paquetes de ataque fueron correctamente detectados y diferenciados en el programa sniffer, reportando un mensaje de alarma por cada uno de ellos y aumentando el estado general de amenaza del sistema a nivel alto. Una vez finalizado el ataque, se constata que el nivel de amenaza se reduce paulatinamente hasta llegar a su mínimo a medida en que el tráfico se fue normalizando. Como se puede ver en la captura del ataque, se detectó también un mensaje IP con dirección de difusión durante la realización de la prueba, que elevó por un momento el nivel de amenaza a crítico. Si bien es cierto que este mensaje no formaba parte de la prueba en sí, sirve para confirmar que el sistema es capaz de procesar amenazas de distinto origen de manera simultánea y efectiva.

2- Cold Restart Attack

Cold Restart Attack es una variante del mismo ataque en la ya analizada versión *warm*, con la salvedad de que el código de función en este caso es 0x0D en lugar de 0x0E. La finalidad del ataque sigue siendo efectuar un ataque de denegación de servicio sobre el RTU, pero en este caso la función supone un reinicio total de las comunicaciones del servidor.

Para comprobar la respuesta del sistema de detección en condiciones similares a las del ataque anterior, se ha generado y enviado un flujo de treinta paquetes con estas características a nuestra RTU de prueba. Al recibirse y procesarse estos paquetes en el destino, la aplicación que simulaba el receptor se ha bloqueado y ha tenido que cerrarse, cortando la comunicación. Al tratarse de una aplicación comercial, este hecho corrobora la facilidad para llevar a cabo y la gravedad que suponen los ataques en DNP3.

Para poder llevar a cabo el ataque evitando fallos en la simulación, se ha introducido un retardo de cien milisegundos entre el envío de cada paquete con función *Cold Restart*, dando tiempo al simulador para procesarlo antes de que el siguiente mensaje llegue y lo encuentre fuera de línea. Con esta salvedad, la comunicación ilegítima no ha supuesto un problema. Al igual que en el caso anterior, se han detectado y analizado los treinta mensajes con esta función y el estado general de alarma ha pasado de bajo a crítico durante la realización del ataque. Dado el gran impacto del ataque sobre los RTU, el que el sistema eleve el estado general de alarma hasta su valor máximo es el resultado esperado.

3- Write Attack

El ataque de escritura o *Write Attack* consiste en el envío de numerosas órdenes de escritura desde un MTU ilegítimo a un RTU. Estas escrituras pueden modificar la configuración, saturar la unidad o provocar errores por desbordamiento de buffer, sin embargo, debemos tener en cuenta que algunas aplicaciones IoT podrían conllevar un

número elevado de mensajes de tipo escritura y por tanto el sistema debe ser menos proclive a levantar alarmas de nivel alto.

Para testar la capacidad de detección de este tipo de ataques, de nuevo lanzamos un flujo de treinta paquetes con la función *write* en capa de aplicación DNP3 desde el script que los fabrica hacia la RTU simulada. En este caso el sistema de monitorización detecta los treinta paquetes fabricados y uno adicional que corresponde a la comunicación legítima que estamos atacando, por lo que de nuevo los resultados coinciden con lo esperado. En cuanto al valor del estado general de alarma, se observa que tras diez paquetes se alcanza un nivel medio que se mantiene hasta que el ataque finaliza, momento a partir del cual se normaliza y vuelve al estado más bajo.

4- Initialize Data Attack

Initialize Data Attack hace referencia a un ciberataque en el que el agresor se vale de la función de inicialización de datos de la capa de aplicación de DNP3 para reinicializar los valores de los objetos en una RTU, pudiendo causar pérdidas de información y toma de decisiones automáticas erróneas en base a información no actualizada. Un flujo elevado de estos mensajes es un signo probable de que se esté produciendo un ataque de estas características.

De manera similar a los casos anteriores, para la comprobación de la funcionalidad del sistema de detección para este ataque, se fabrican y envían treinta paquetes con esta función en la cabecera de la capa de aplicación. Se comprueba que los treinta paquetes han sido correctamente detectados por el sniffer y que se han reportado por pantalla de manera correcta. Se puede observar también que el estado general de alarma del sistema ha alcanzado el nivel máximo a partir del décimo paquete y se ha mantenido estable hasta la finalización del ataque, cuando ha comenzado a decrecer.

5- App Function Termination Attack

El ataque de terminación de función en capa de aplicación provoca la finalización de los procesos en ejecución en la RTU. De manera legítima un maestro puede utilizar esta función para cerrar servicios innecesarios en los esclavos, pero un atacante puede aprovechar esta función para generar una inundación de mensajes de terminación y bloquear puntual o indefinidamente la conexión.

Para comprobarlo se fabrican y lanzan al receptor treinta paquetes con esta función en capa de aplicación. Al llegar el primero de ellos la aplicación se paraliza momentáneamente y se desconecta, rechazando el envío en el emisor de los veintinueve paquetes fabricados restantes. Tras una pequeña pausa, se vuelve a intentar el envío de otros treinta paquetes, y esta vez sí que es posible la recepción completa por parte de receptor. En los resultados finales se puede observar como el sistema de monitorización ha detectado el paquete que causó la primera parada y los otros treinta del segundo envío. Respecto al estado general de alarma del sistema, el resultado es similar al del ataque de inicialización de datos ya visto, alcanzando el valor máximo a partir del décimo paquete y manteniéndolo hasta la finalización del ataque. La explicación de esto es que ambos ataques tienen la misma ponderación de riesgo en el archivo de configuración, por lo que el sistema general de alarma responde de manera idéntica al ser un flujo del mismo número de paquetes.

6- Delete File Attack

Este ataque supone el borrado de los archivos del esclavo. Al igual que en el resto de ataques, es una función cuyo uso repetido puede interpretarse como un ciberataque, ya que puede provocar pérdidas de información y bloqueos. Es especialmente importante tener en cuenta que el empleo de este ataque conlleva generalmente el borrado de los *logs* de tráfico que registra el RTU, lo que puede utilizarse por una persona malintencionada como un método para borrar sus huellas.

Al igual que en los otros casos ya estudiados, al generar y enviar el flujo de paquetes de estas características hacia el receptor para su testeo, se logra identificar y reportar por separado cada uno de los ataques. El sistema general de alarma alcanza el valor crítico a mitad del ataque, y después disminuye.

5.4 Conclusiones

Los resultados de las distintas pruebas realizadas sobre el sistema de detección durante la fase de validación permiten afirmar que cumple los requisitos funcionales y no funcionales para los que se había diseñado. El detalle de los distintos requisitos puede encontrarse en el Anexo A.

A partir de las pruebas que han sido realizadas sobre distintos ataques, podemos afirmar que se cumplen los requisitos funcionales de monitorización de tráfico en la interfaz (RF1), de procesamiento de tráfico paquete a paquete sin pérdidas a tasas medias (RF2) y de procesamiento de paquetes y extracción de datos de interés para la aplicación (RF4 y RF3, respectivamente). Así mismo, también se cumplen los requisitos de análisis de datos en tiempo real y reconocimiento de ataques (RF5, RF6, RF7).

En las capturas del Anexo C se incluyen los resultados de la ejecución del programa, en las que vemos que se muestra información de los ataques acontecidos en tiempo real (RF9, RF11), alertas de cambio del estado general del sistema (RF8,RF10) y un resumen final con los datos más relevantes derivados de la ejecución del programa (RF12).

Respecto a los requisitos no funcionales, el sistema de monitorización ha demostrado la capacidad de procesamiento fiable de paquetes (RNF1, RNF3), ser estable en el tiempo (RNF2), proporcionar datos útiles a un operador no necesariamente experto en un tiempo razonable (RNF7,RNF6) y tener un muy coste de implementación muy bajo (RNF4).

6 Conclusiones y trabajo futuro

6.1 Conclusiones

El objetivo de este Trabajo Fin de Grado era el desarrollo y la implementación de un sistema de monitorización de tráfico anómalo en el contexto de la Internet de las Cosas, que permitiera detectar ataques sobre este nuevo paradigma de comunicación.

Para realizar este proyecto se ha llevado a cabo un estudio exhaustivo sobre el estado del arte en temática de IoT, SCADA, DNP3 y detección de ataques en ámbito IDS. Durante la investigación se ha adquirido una base de conocimiento en estas materias que ha permitido entender nuevos conceptos, ampliar otros ya conocidos y producir una panorámica muy valiosa acerca de la complejidad de las redes IoT y la importancia de garantizar férreas medidas de ciberseguridad sobre ellas.

Uno de los pilares fundamentales ha sido el desarrollo de la aplicación de detección de tráfico anómalo, el cual ha supuesto la necesidad de la obtención de conocimiento y estudio a nivel bajo del modelo de comunicación de DNP3 con el fin de entender el funcionamiento del mismo. El desarrollo software en lenguaje Python ha representado también una parte vital del proyecto, por lo que se ha tenido que hacer un esfuerzo en obtener conocimiento técnico y operativo sobre el desarrollo de herramientas de red en dicho lenguaje. Así mismo, para llevar a cabo la depuración de errores de la aplicación y la validación de resultados, ha sido muy importante la preparación de un entorno de trabajo para simular un sistema de comunicación de estas características, mediante el que se ha adquirido experiencia acerca de la virtualización de sistemas y la instalación de diversas herramientas no comerciales.

El Trabajo Fin de Grado ha conllevado también la consolidación del conocimiento adquirido en muchas materias cursadas durante la carrera. Los conceptos de redes, topologías y modelos de comunicación estudiados en las asignaturas *Arquitectura de Redes I* y *Arquitectura de Redes II* se han visto reforzados durante el proceso de entendimiento del protocolo DNP3. Asignaturas como *Programación I*, *Programación II* y *Redes Multimedia* han sentado las bases técnicas necesarias para el desarrollo software del sistema de detección, siendo especialmente importante esta última por su focalización en el lenguaje Python. Por último, asignaturas de la rama de Electrónica tales como *Sistemas de Control*, *Dispositivos Integrados Especializados*, *Aritmética para Procesamiento de la Señal* y *Tecnología Electrónica de Sistemas* han servido para entender en un contexto mucho más amplio los diferentes tipos de dispositivos electrónicos que componen las redes IoT, aportando una visión de alto nivel sobre este tipo de implementaciones.

En vista de los buenos resultados obtenidos en las pruebas de validación, se concluye el trabajo habiendo alcanzado los diferentes objetivos propuestos y desarrollados en los capítulos anteriores. La experiencia y los conocimientos adquiridos durante el desarrollo del proyecto resultan muy enriquecedores y no cabe duda de que podrán ser aprovechados en el futuro.

6.2 Trabajo futuro

A pesar de que se considera que el proyecto se ha finalizado de manera satisfactoria, se identifican posibles líneas de trabajo que amplíen este estudio, mejoren las herramientas creadas y extiendan aspectos que no se hayan llegado a tratar, como por ejemplo:

- Realización de pruebas en un escenario más complejo, bien sea a través de un software de simulación plenamente funcional o una red real. Dichas pruebas deberían incluir un test de *fuzzing* para comprobar la capacidad de detección del sistema a este tipo de ataques.

- Mejora del sistema de gestión de alarmas, dotándolo de mecanismos de suavizado exponencial e histéresis para evitar transiciones bruscas entre estados.

- Ampliación de la funcionalidad del sistema de visualización de para que sea capaz de mostrar gráficas en tiempo real que aumenten la legibilidad.

- Aumento de la capacidad de procesamiento de paquetes por segundo de la aplicación mediante el desarrollo de un *wrapper* de la misma en un lenguaje de programación más rápido.

Como parte del planteamiento del trabajo futuro y en firme apoyo a las ideas y principios del software libre, todo el código desarrollado a lo largo del proyecto se ha publicado en el repositorio GitHub del laboratorio de investigación HPCN de la Universidad Autónoma de Madrid. Los links a través de los cuáles se puede acceder al proyecto son los siguientes:

- Sistema de detección de tráfico anómalo:
  https://github.com/hpcn-uam/DNP3-Attack-Detection-System

- Generador de tráfico DP3Crafter:
  https://github.com/hpcn-uam/DNP3Crafter

7 Referencias bibliográficas

[1] EVANS, Dave. *The Internet of Things. How the Next Evolution of the Internet Is Changing Everything*. Cisco Systems. Abril de 2011

[2] ATZORI, Luigi; IERA, Antonio; et al. *The Internet of Things: A Survey*, 2010

[3] ASHTON, Kevin. *That 'Internet of Things' thing*. RFID Journal, 2009

[4] MIORANDI, Daniele; et al. *Internet of Things: Vision, Applications and Research Challenges*. Ad Hoc Networks, 2012

[5] WEISER, Mark; GOLD, Rich; et al. *The origins of ubiquitous computing research at PARC in the late 1980*. IBM Systems Journal, Vol 38, Nº4, 1999

[6] GUBBI, Jayavardhana; BUYYA, Rajkumar; et al. *Internet of Things (IoT): A vision, architectural elements, and future directions*. Future Generation Computer Systems. Elsevier. Septiembre de 2013.

[7] RIAHI, Arbia; NATALIZIO, Enrico; et al. *A systemic and cognitive approach for IoT security*. International Conference on Computing, Networking and Communications, 2014

[8] National Communications System. *Supervisory Control and Data Acquisition (SCADA) Systems*. Arlington, EEUU. 2004

[9] DANEELS, Axel; SALTER, Wayne. *What is SCADA?*. International Conference on Accelerator and Large Experimental Physics Control Systems. Trieste, 1999

[10] ZIMMER, Dave, RHODES, Duncan. *Human-machine Interfaces. Mill requirements for drive system trending HMIs*. IEEE Industry Applications Magazine, Marzo/Abril de 2006

[11] Triangle Microworks, Inc. *DNP3 Overview*. Raleigh, North Carolina.

[12] CLARKE, Gordon; REYNDERS, Deon; et al. Practical Modern SCADA Protocols: DNP3, IEC 60870.5 and related systems. Elsevier, 2004.

[13] ALI, Mohammed. *DNP3 Analysis and Implementations*. University of Khartoum. 2014.

[14] MESERVE, Jeann. *Mouse click could plunge city into darkness, experts say, CNN.com* (www.cnn.com/2007/US/09/27/power.at.risk/index.html), 27 de septiembre de 2007.

[15] FOVINO, Igor; CARCANO, Andrea, et al. *Modbus/DNP3 State-based Intrusion Detection System*. 2010 24th IEEE International Conference on Advanced Information Networking and Applications. 2010.

[16] J. COX, Kerry; GERG, Cristopher. *Managing Security with Snort & IDS Tools*. O'Reilly Media, Inc. 2004

[17] RODOFILE, Nicholas; RADKE, Kenneth; FOO, Ernest. *Real-Time and Interactive Attacks on DNP3 Critical Infraestructure Using Scapy*. AISC 2015. Australia, 2015.

[18] EAST, Samuel et al. *A Taxonomy Of Attacks on the DNP3 Protocol*. Critical Infraestructure Protection III, IFIP AICT 311, pp. 67-81, 2009

[19] GARCÍA, Fernando. *Análisis de las normas de seguridad para los controles, las comunicaciones y otros equipos críticos de la red de energía*. Proyecto Fin de Carrera. Escuela Politécnica Superior. Universidad Autónoma de Madrid. Mayo de 2014

[20] Industrial Ethernet Book Issue. *The protocols that are driving smart grid interoperability*. IEB Media GBR, noviembre de 2015.

[21] BIONDI, Philippe. *Scapy: explore the net with new eyes*. EADS Corporate Research Center, SSI department. Francia, 2015

[22] MAXWELL, Adam. *The very unofficial dummies guide to Scapy*. itgeekchronicles.co.uk/

[23] Automatak, Inc. *What is fuzzing?*. Aegis Platform 0.2.0 documentation. https://www.automatak.com/aegis/docs/intro/fuzzing.html

Anexos

Anexo A: Tabla de requisitos del Trabajo Fin de Grado

A.1 – Requisitos funcionales

Tabla 2 - Requisitos funcionales

Requisito	Modulo	Descripción
RF1	Captura del tráfico	Deberá ser capaz de monitorizar todo el tráfico de red en una interfaz.
RF2	Captura del tráfico	Deberá ser capaz de procesar el tráfico paquete a paquete, evitando pérdidas en el *kernel*.
RF3	Captura del tráfico	Deberá poder extraer valores de tráfico que sean de interés para la aplicación.
RF4	Captura del tráfico	Deberá ser capaz de procesar los paquetes de manera rápida y eficiente.
RF5	Mecanismo de alarma	Deberá contar con un mecanismo que analice los datos obtenidos en tiempo real.
RF6	Mecanismo de alarma	Deberá ser capaz de reconocer patrones de ataque basados en parámetros estadísticos o en firmas de ataques complejos del protocolo DNP3.
RF7	Mecanismo de alarma	Deberá poder valerse de operaciones estadísticas para hacer comprobaciones.
RF8	Mecanismo de alarma	Deberá poseer un sistema de máquina de estados que actualice el estado general de defensa del sistema cuando sea necesario.
RF9	Visualización de datos en tiempo real	Deberá disponer de un sistema que muestre por pantalla la información relativa a cada evento de ataque que suceda en tiempo real.
RF10	Visualización de datos en tiempo real	Complementariamente a RF9, deberá alertar también de los cambios en el estado general de la defensa del sistema.
RF11	Visualización de datos en tiempo real	Especificará información adicional en cada evento, como la severidad o el instante temporal del ataque.
RF12	Resumen final	Deberá mostrar un resumen numérico de los eventos más importantes ocurridos durante la monitorización.
RF13	Utilidad de configuración	Deberá proveerse un archivo de configuración en formato *.ini* que permita al usuario alterar los parámetros que definen las alarmas.
RF14	Utilidad de configuración	Deberá existir documentación que acompañe a cada variable del archivo configuración, indicando el parámetro al que hacen referencia y el valor por defecto.

A.2 – Requisitos no funcionales

Tabla 3 - Requisitos no funcionales

Requisito	Nombre	Descripción
RNF1	Fiabilidad	El sistema debe obtener datos y métricas lo más precisas posibles dada la necesidad de procesar alarmas en tiempo real.
RNF2	Estabilidad	El conjunto de la aplicación deberá ser capaz de funcionar durante periodos largos de tiempo dando cobertura en tiempo real.
RNF3	Rendimiento	El sistema debe disponer de la capacidad para procesar los paquetes de manera eficaz y soportar tasas de mensajes medias.
RNF4	Economía	El coste de la instalación del sistema en una aplicación de Internet de las Cosas debe ser lo más bajo posible.
RNF5	Inmediatez	Los datos obtenidos a través de la ejecución del programa deben ser fácilmente comprensibles para el operador.
RNF6	Sencillez	El sistema deberá ser intuitivo, de modo que pueda ser utilizado por un técnico con pocas aptitudes informáticas.
RNF7	Extensibilidad	El código deberá estar correctamente documentado y regirse por los principios de alta cohesión y bajo acoplamiento para que sea fácil continuar su desarrollo por otro programador.

Anexo B: Entorno de trabajo

En este anexo se describen los medios utilizados como entorno de trabajo para la realización del proyecto. Todo el conjunto de software y máquinas virtuales que se detallan a continuación corren sobre un equipo de sobremesa con un procesador AMD Phenom 2 X6 1055T con 8GB de memoria RAM.

B.1 - Sistema anfitrión

El sistema anfitrión sobre el que se desarrolla el proyecto es Windows 8.1. Sobre él se ejecutan las máquinas virtuales de los otros sistemas implicados en el proyecto además de parte del software de simulación y testeo. La virtualización de los sistemas se efectúa mediante el programa VirtualBox.

B.2 - Máquina virtual I. Receptor.

La simulación de la unidad terminal remota se efectúa a través de una instancia de Axon Test corriendo sobre un sistema Linux Mint 17.1 virtualizado. Es en esta máquina donde se ejecuta también el sistema de detección de tráfico anómalo, *sniff.*py, de manera que se pueda ser capaz de procesar la comunicación DNP3 del lado del receptor.

Como ya se ha hablado durante el capítulo del desarrollo de la solución, el sistema de monitorización desarrollado requiere de la herramienta Scapy para funcionar. Esto determina el motivo de utilizar Linux en esta parte del entorno, ya que Scapy no funciona de manera completa en entornos Windows.

Para ser capaces de ejecutar Axon Test en Linux Mint, el cual no es soportado de manera oficial, se ha optado por configurar una capa de virtualización de Wine con .NET Framework 3.5.

Además del software ya mencionado, esta máquina virtual dispone de los prerrequisitos del sistema de detección -biblioteca pcap y Python 2.7- y de Wireshark, este último utilizado para comprobar que los procesos de sniffing del sistema desarrollado eran precisos.

B.3 - Máquina virtual II. Emisor atacante.

La simulación del emisor atacante se realiza a través de la ejecución del script *DNP3Crafter.py* creado durante el proyecto sobre una máquina virtual con sistema operativo Linux Mint 17.1.

Como el módulo *DNP3Crafter* se basa en sockets de Python, se ha hecho necesario instalar también Python 2.7 y la biblioteca pcap como prerrequisitos.

Anexo C: Software utilizado

C.1 - Scapy

Scapy es una herramienta interactiva de manipulación de paquetes, capaz de crear o decodificar paquetes de un gran número de protocolos, enviarlos o recibirlos por una interfaz y hacer capturas de tráfico. El programa, desarrollado por Philippe Biondi, está programado sobre Python y provee de un intérprete de aplicación sobre el que se puede trabajar directamente además de bibliotecas para desarrollar aplicaciones o scripts. [9]

Esta aplicación trabaja de forma nativa sobre sistemas Linux y se basa en bibliotecas ampliamente conocidas como libpcap y lidnet con su correspondiente *wrapper* en Python. Aunque puede hacerse funcionar en sistemas Windows siguiendo algunos pasos adicionales, es poco recomendable porque presenta varios bugs que no han sido corregidos.

El factor que diferencia a Scapy de otros programas del ámbito de la manipulación de paquetes es que no está orientado a una aplicación específica, sino que permite al desarrollador construir el paquete de la manera que este quiera, incluso aunque a priori el paquete generado no tenga sentido. Esta flexibilidad aporta un aliciente para probar protocolos poco conocidos, evitando tener que buscar simuladores específicos.

El fin de esta herramienta es proveer al usuario de capacidad de decodificación de paquetes en lugar de la interpretación de los mismos. Esto es así porque el autor considera que en la decodificación de los paquetes, las herramientas comunes sesgan información que podría ser útil para el usuario, por lo que se considera que la herramienta debe limitarse a decodificar los mensajes y dejar que sea el usuario el que extraiga de ellos la información que crea valiosa.

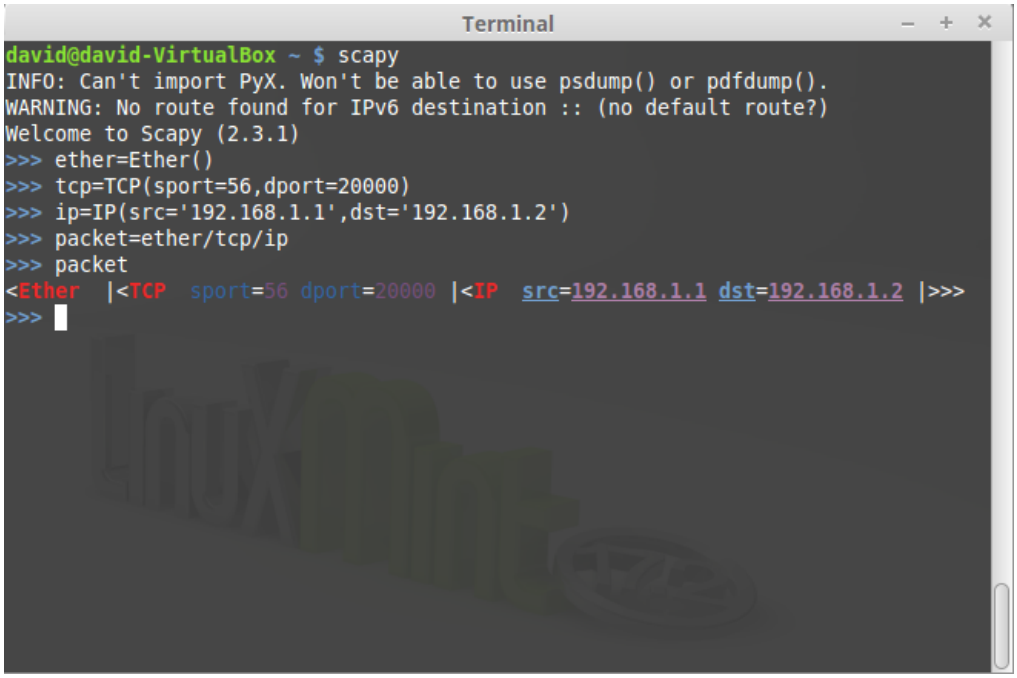

Figura 10 – Creación de un paquete con Scapy

Scapy está pensado para permitir el diseño rápido de paquetes de una manera flexible, sin valores restringidos y sin límite de combinaciones. Para ello, el intérprete funciona de manera similar a un DSL donde cada paquete se divide en capas (Ethernet, IP, TCP, UDP…) que pueden apilarse para generar un conjunto.

Cada una de estas capas posee una serie de campos del protocolo al que representan. El usuario puede modificar los valores de estos campos para adecuarlos a su intención concreta, pero no será necesario asignar un valor a cada campo puesto que ya contienen ciertos parámetros fijados por defecto, con lo que la velocidad de creación de un paquete es muy alta. En caso de modificar un campo, el valor por defecto se sobrescribe con la elección del usuario.

A pesar de que Scapy soporta de manera nativa gran cantidad de protocolos, ya se ha mencionado que uno de los puntos fuertes de la herramienta es la capacidad de desarrollar aplicaciones nuevas con relativa facilidad. Podemos añadir nuevos estándares simplemente definiendo una nueva clase en Python, ya que hay una analogía directa entre los campos de las capas de Scapy y los elementos que componen su clase.

Además de generar un paquete a partir de la unión de capas de Scapy, también es posible añadir carga a un paquete directamente en crudo como una cadena de bytes que se considerará carga útil.

El uso de Scapy ha sido fundamental para la creación y decodificación de paquetes en este proyecto, habiéndose ampliado la funcionalidad básica de la herramienta a través de la biblioteca *DNP3_Lib* proporcionada por Nicholas Rodofile [11], la cual permite el tratamiento directo de paquetes DNP3 añadiendo otra nueva capa de abstracción.

C.2 - Axon test

Simulador de tráfico SCADA que permite enviar y recibir tráfico específico de los protocolos DNP3, IEC 80670-5-101, IEC 80670-5-104 y Modbus a través de la virtualización de MTUs y RTUs.

Axon Test funciona sobre sistemas Windows y puede descargarse de manera gratuita en la página web de su desarrollador, http://www.axongroup.com.co/axon_descargas.php, disponiendo de una versión de prueba de cuarenta días que al expirar deriva en la versión *freeware* que se ha utilizado en este Trabajo Fin de Grado.

Entre las limitaciones con las que nos encontramos en la versión gratuita destaca el no poder incluir más de un dispositivo por cada instancia del programa, lo que imposibilita la simulación local. Además de esto, no se dispone de la opción de guardar los proyectos ni de obtener ayuda del soporte técnico y las opciones de gestión de entradas analógicas y digitales de la simulación son muy reducidas.

A pesar de estas restricciones, Axon Test en su versión gratuita ha sido suficiente para simular una conexión DNP3 básica entre maestro y receptor y poder así validar los resultados de detección de parámetros en lo que al tráfico legítimo se refiere.

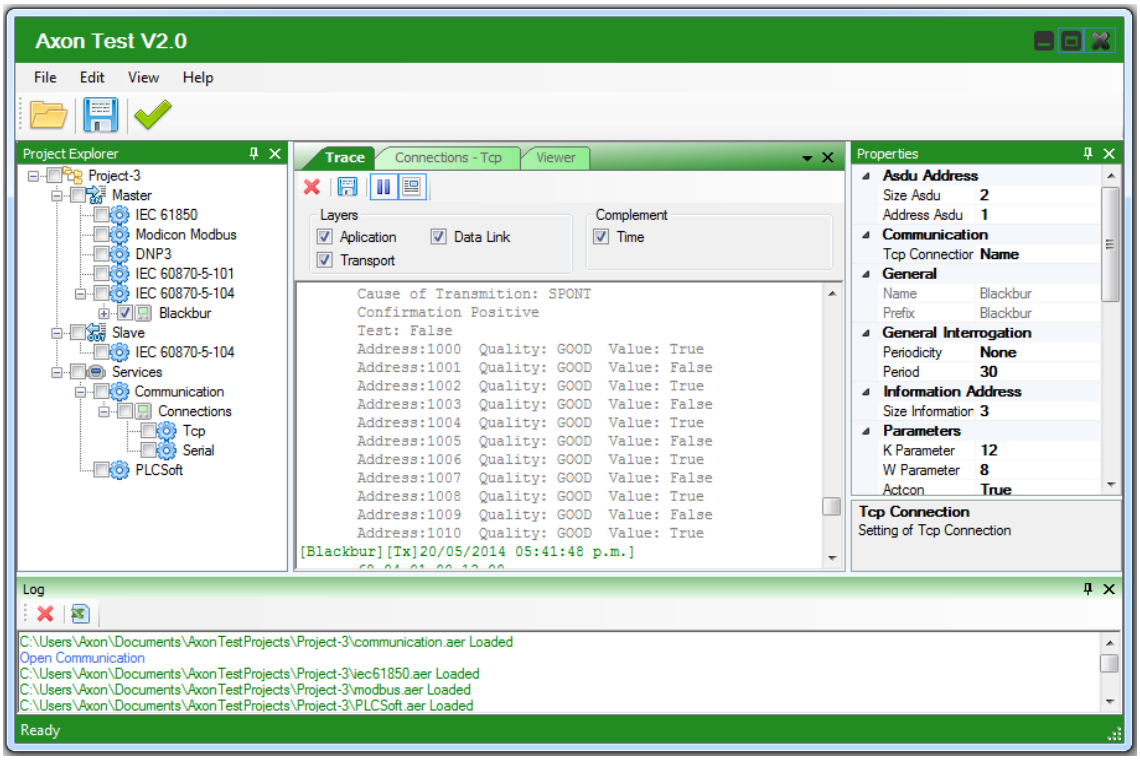

Figura 11 – Interfaz de Axon Test (www.axongroup.com)

C.3 - Aegis (versión Freeware)

Herramienta desarrollada por Automatak® para la realización de test de *fuzzing* sobre protocolos SCADA. La versión *freeware* solo puede ser empleada en DNP3 y carece de interfaz gráfica, pero dispone de gran cantidad de opciones para configurar los ataques a través de parámetros de entrada del programa, algunos de los cuáles se describen a continuación:

Tabla 4 - Parámetros de entrada de Aegis

Argumento	Descripción
-mid	Identifica el protocolo sobre el que se van a desarrollar las pruebas. En la versión freeware está restringido a DNP3, pero su versión comercial incluye la capacidad de testear Modbus.
-pid	Identifica la prueba de fuzzing que se va a realizar. Se puede elegir entre fuzzing de la capa de enlace (*lfuzz*), fuzzing de las funciones de transporte (*tfuzz*), fuzzing de la cabecera del nivel de aplicación y de sus códigos de función (*ahfuzz*) y fuzzing profundo de la capa de aplicación (*aofuzz*).
-host	Determina la dirección IP del objetivo. Si no se especifica un valor, se fija 127.0.0.1.
-port	Determina el puerto objetivo del test. Si no se especifica un valor, se fija el puerto 20000.
-listen	Escucha en el puerto previamente especificado en lugar de empezar un test de fuzzing.
-seed	Fija una semilla específica para ejecutar dos veces el mismo test.
-dest	Permite fijar una dirección de destino de la capa de enlace.

-src	Fija la dirección de origen de la capa de enlace. Si no se fija, se toma la dirección de origen real. Esta opción es importante a la hora de enmascarar la fuente del test.
-master	Configura el programa para llevar a cabo tests de fuzzing sobre unidades maestro o MTU. Si no se especifica un valor, la configuración se ajusta para testear RTUs.
-retries	Delimita el número de intentos de conexión fallidos antes de abortar la ejecución del test.
-linktimeout	Delimita el tiempo máximo a esperar para recibir una trama del objetivo antes de abortar la ejecución del test.
-fill	Permite determinar el valor de los bytes que se van a rellenar aleatoriamente en la generación aleatoria de paquetes durante el test. Si no se especifica un valor, se fija 0xFF.

Durante la fase de testeo de este Trabajo Fin de Grado se trató de llevar a cabo un test de *fuzzing* con Aegis sobre el receptor simulado con el fin de probar el comportamiento del sistema de detección de tráfico anómalo. Esta prueba no dio resultado debido a que el receptor DNP3, simulado mediante Axon Test, no contestaba de manera apropiada a los *health checks* que demanda este *fuzzer*.

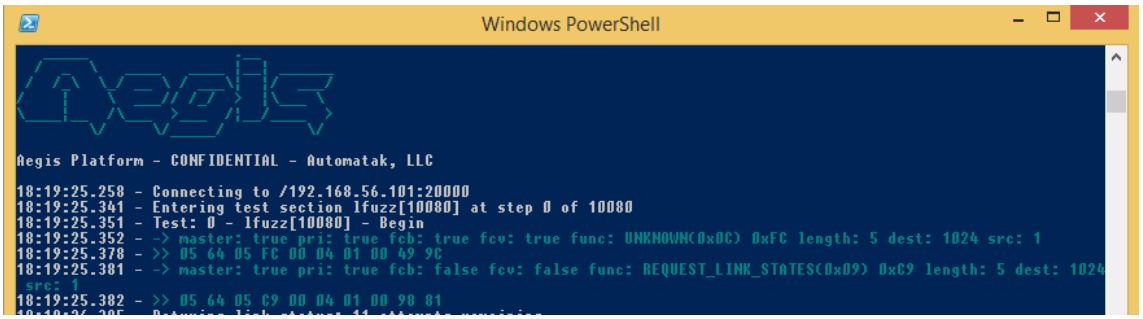

Figura 12 - Aegis efectuando un test de fuzzing sobre un host

Anexo D: Capturas de ataque

D.1 - Prueba de rendimiento ante *flood* de escrituras

Tabla 5 - Prueba de rendimiento de la aplicación ante flood de paquetes

Paquetes enviados	Retardo entre paquetes en el emisor	Paquetes procesados en recepción	Porcentaje de pérdidas
500	100 ms	500	0%
500	75 ms	500	0%
500	50 ms	500	0%
500	40 ms	500	0%
500	35 ms	446	10,8%
500	30 ms	386	22,8%
500	25 ms	340	32%
500	20 ms	274	45,2%

D.2 - IP Range out of bounds

```
       Detected      Severity      Time
       --------      --------      ----
ALERT: AnormalIPRange        High    2015-11-22 13:20:54.236687
ALERT: AnormalIPRange        High    2015-11-22 13:20:55.486945
ALERT: AnormalIPRange        High    2015-11-22 13:20:57.811448
ALERT: AnormalIPRange        High    2015-11-22 13:20:57.812085
ALERT: AnormalIPRange        High    2015-11-22 13:20:58.313530
ALERT: AnormalIPRange        High    2015-11-22 13:20:58.314147
ALERT: AnormalIPRange        High    2015-11-22 13:20:58.814675
ALERT: AnormalIPRange        High    2015-11-22 13:20:58.815430
ALERT: AnormalIPRange        High    2015-11-22 13:21:08.832156
ALERT: AnormalIPRange        High    2015-11-22 13:21:08.832774
EVENT: Current alarm status is: Medium     2015-11-22 13:21:08.832831
ALERT: AnormalIPRange        High    2015-11-22 13:21:09.333257
ALERT: AnormalIPRange        High    2015-11-22 13:21:09.333946
ALERT: AnormalIPRange        High    2015-11-22 13:21:09.834762
ALERT: AnormalIPRange        High    2015-11-22 13:21:09.835578
ALERT: AnormalIPRange        High    2015-11-22 13:21:20.380766
ALERT: AnormalIPRange        High    2015-11-22 13:21:20.433811
ALERT: AnormalIPRange        High    2015-11-22 13:21:24.855480
ALERT: AnormalIPRange        High    2015-11-22 13:21:24.856362
ALERT: AnormalIPRange        High    2015-11-22 13:21:24.857054
ALERT: AnormalIPRange        High    2015-11-22 13:21:24.858982
ALERT: WriteFunctionDetected         Normal 2015-11-22 13:21:24.859303
ALERT: AnormalIPRange        High    2015-11-22 13:21:24.860048
ALERT: AnormalIPRange        High    2015-11-22 13:21:24.893770
ALERT: AnormalIPRange        High    2015-11-22 13:21:24.932557
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.025816
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.064208
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.082679
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.123345
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.235051
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.236333
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.444951
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.493442
ALERT: AnormalIPRange        High    2015-11-22 13:21:25.497066
ALERT: AnormalIPRange        High    2015-11-22 13:21:30.438101
ALERT: AnormalIPRange        High    2015-11-22 13:21:30.490498

Sniffer stopped by keystroke.

#### Final results ####
Time elapsed: 46.469 seconds.
Packets sniffed: 40
- IP packets: 34 Percent: 85.0 %
---- TCP packets: 27 Percent: 67.5 %
---- UDP packets: 7 Percent: 17.5 %
DNP3 messages: 6

DNP3 Application Layer Function Codes:
READ : 1
WRITE : 1
ENABLE_UNSOLICITED : 1
RESPONSE : 3
```

D.3 - Warm Restart Attack

```
       Detected       Severity       Time
       --------       --------       ----
ALERT: WriteFunctionDetected        Normal 2015-11-21 20:53:35.770586
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:46.487581
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:46.589467
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:46.691545
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:46.792666
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:46.893914
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:46.994897
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.095477
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.196216
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.296628
EVENT: Current alarm status is: Medium     2015-11-21 20:53:47.296801
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.397536
EVENT: Current alarm status is: High       2015-11-21 20:53:47.397733
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.498468
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.599417
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.700149
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.801493
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:47.902158
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.002489
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.102934
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.206843
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.305449
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.406459
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.507982
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.608830
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.710131
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.811400
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:48.913193
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:49.013351
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:49.114073
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:49.214600
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:49.315354
ALERT: ResetFunctionAttack   High    2015-11-21 20:53:49.416453
ALERT: IPBroadcastMessage    High    2015-11-21 20:53:59.502159
EVENT: Current alarm status is: Critical   2015-11-21 20:53:59.502185
EVENT: Current alarm status is: High       2015-11-21 20:53:59.502290
EVENT: Current alarm status is: Medium     2015-11-21 20:54:16.844977
EVENT: Current alarm status is: Low        2015-11-21 20:54:57.238090

Sniffer stopped by keystroke.

#### Final results ####
Time elapsed: 105.218 seconds.
Packets sniffed: 137
- IP packets: 121 Percent: 88.3211678832 %
---- TCP packets: 93 Percent: 67.8832116788 %
---- UDP packets: 36 Percent: 26.2773722628 %
DNP3 messages: 44

DNP3 Application Layer Function Codes:
READ : 5
WRITE : 1
WARM_RESTART : 30
ENABLE_UNSOLICITED : 1
RESPONSE : 7
```

D.4 - Cold Restart Attack

```
       Detected      Severity      Time
       --------      --------      ----
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:21:59.333699
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:21:59.434914
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:21:59.534787
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:21:59.637008
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:21:59.737576
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:21:59.838255
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:21:59.939867
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.039426
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.140120
EVENT: Current alarm status is: Medium     2015-11-22 16:22:00.140334
EVENT: Current alarm status is: High       2015-11-22 16:22:00.141051
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.240786
EVENT: Current alarm status is: Critical   2015-11-22 16:22:00.241056
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.341307
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.441284
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.542338
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.642362
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.743564
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.844062
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:00.943800
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.046109
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.146538
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.246439
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.347853
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.448611
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.549152
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.650396
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.749998
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.850951
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:01.952417
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:02.052906
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:02.154021
ALERT: ColdResetFunctionAttack      Critical      2015-11-22 16:22:02.256016
ALERT: WriteFunctionDetected        Normal 2015-11-22 16:22:11.509464
EVENT: Current alarm status is: High       2015-11-22 16:22:12.187491
EVENT: Current alarm status is: Medium     2015-11-22 16:22:32.719739
EVENT: Current alarm status is: Low        2015-11-22 16:22:32.725588

Sniffer stopped by keystroke.

#### Final results ####
Time elapsed: 238.249 seconds.
Packets sniffed: 199
- IP packets: 171 Percent: 85.9296482412 %
---- TCP packets: 133 Percent: 66.8341708543 %
---- UDP packets: 54 Percent: 27.135678392 %
DNP3 messages: 57

DNP3 Application Layer Function Codes:
READ : 12
WRITE : 1
COLD_RESTART : 30
ENABLE_UNSOLICITED : 1
RESPONSE : 13
```

D.5 - Write Attack

```
       Detected      Severity      Time
       --------      --------      ----
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:06.190473
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:11.740969
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:11.842429
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:11.943946
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.045522
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.146635
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.246782
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.347728
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.447881
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.548587
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.649361
EVENT: Current alarm status is: Medium    2015-11-22 16:55:12.649631
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.750206
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.850220
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:12.951063
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.051345
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.152176
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.254092
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.355498
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.458650
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.558532
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.658814
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.760757
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.861490
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:13.963479
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:14.063036
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:14.163870
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:14.265545
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:14.366693
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:14.467269
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:14.569249
ALERT: WriteFunctionDetected       Normal 2015-11-22 16:55:14.669423
EVENT: Current alarm status is: Low       2015-11-22 16:56:07.329670

Sniffer stopped by keystroke.

#### Final results ####
Time elapsed: 103.274 seconds.
Packets sniffed: 109
- IP packets: 107 Percent: 98.1651376147 %
---- TCP packets: 95 Percent: 87.1559633028 %
---- UDP packets: 12 Percent: 11.0091743119 %
DNP3 messages: 44

DNP3 Application Layer Function Codes:
READ : 5
WRITE : 31
ENABLE_UNSOLICITED : 1
RESPONSE : 7
```

D.6 - Initialize Data Attack

```
       Detected       Severity       Time
       --------       --------       ----
ALERT: WriteFunctionDetected        Normal 2015-11-22 17:00:05.544446
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:12.433580
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:12.533824
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:12.635580
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:12.736904
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:12.838439
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:12.938986
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.040241
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.140296
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.241637
EVENT: Current alarm status is: Medium     2015-11-22 17:00:13.241827
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.343939
EVENT: Current alarm status is: High       2015-11-22 17:00:13.344109
EVENT: Current alarm status is: Critical   2015-11-22 17:00:13.344851
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.443967
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.544280
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.645739
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.745594
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.846179
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:13.946762
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.047122
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.147921
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.249011
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.349109
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.449505
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.550138
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.651785
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.753427
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.854341
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:14.954406
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:15.055059
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:15.156879
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:15.257327
ALERT: InitializeDataFunctionAttack        High   2015-11-22 17:00:15.359047
EVENT: Current alarm status is: High       2015-11-22 17:00:26.511128
EVENT: Current alarm status is: Medium     2015-11-22 17:01:06.773706
EVENT: Current alarm status is: Low        2015-11-22 17:01:26.898169
ALERT: IPBroadcastMessage    High   2015-11-22 17:02:05.295371
EVENT: Current alarm status is: Medium     2015-11-22 17:02:05.295403

Sniffer stopped by keystroke.

#### Final results ####
Time elapsed: 135.967 seconds.
Packets sniffed: 123
- IP packets: 115 Percent: 93.4959349593 %
---- TCP packets: 99 Percent: 80.487804878 %
---- UDP packets: 16 Percent: 13.0081300813 %
DNP3 messages: 48

DNP3 Application Layer Function Codes:
READ : 7
WRITE : 1
INITIALIZE_DATA : 30
ENABLE_UNSOLICITED : 1
RESPONSE : 9
```

D.7 - Application Function Termination Attack

```
        Detected       Severity       Time
        --------       --------       ----
ALERT: WriteFunctionDetected        Normal  2015-11-22 17:04:28.360156
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:33.880506
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:38.224794
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:38.323527
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:38.424680
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:38.525842
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:38.626491
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:38.726608
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:38.827754
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:38.927357
EVENT: Current alarm status is: Medium      2015-11-22 17:04:38.927484
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.030045
EVENT: Current alarm status is: High        2015-11-22 17:04:39.030170
EVENT: Current alarm status is: Critical    2015-11-22 17:04:39.030922
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.131331
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.233077
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.334148
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.435210
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.535429
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.637063
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.738116
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.838314
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:39.940335
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.040762
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.141204
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.241987
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.343056
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.445753
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.547848
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.648456
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.748456
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.849238
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:40.949626
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:41.050933
ALERT: AppTerminationFunctionAttack         Critical       2015-11-22 17:04:41.150971
ALERT: WriteFunctionDetected        Normal  2015-11-22 17:04:48.413582
EVENT: Current alarm status is: High        2015-11-22 17:04:48.435226
EVENT: Current alarm status is: Medium      2015-11-22 17:04:50.234070
EVENT: Current alarm status is: Low         2015-11-22 17:05:11.923696

Sniffer stopped by keystroke.

#### Final results ####
Time elapsed: 144.004 seconds.
Packets sniffed: 162
- IP packets: 139 Percent: 85.8024691358 %
---- TCP packets: 117 Percent: 72.2222222222 %
---- UDP packets: 34 Percent: 20.987654321 %
DNP3 messages: 49

DNP3 Application Layer Function Codes:
READ : 6
WRITE : 2
STOP_APPL : 31
ENABLE_UNSOLICITED : 2
RESPONSE : 8
```

D.8 - Delete File Attack

```
Sniffing process started. To stop it, press Ctrl+C
       Detected      Severity      Time
       --------      --------      ----
ALERT: WriteFunctionDetected       Normal 2015-11-22 17:12:21.542056
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:32.333459
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:32.434770
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:32.535128
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:32.636323
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:32.736496
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:32.836771
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:32.940239
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.039069
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.140611
EVENT: Current alarm status is: Medium     2015-11-22 17:12:33.140776
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.241892
EVENT: Current alarm status is: High       2015-11-22 17:12:33.242010
EVENT: Current alarm status is: Critical   2015-11-22 17:12:33.242780
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.341951
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.442145
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.543168
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.643553
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.744379
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.845292
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:33.945104
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.046677
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.147802
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.249261
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.351001
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.453066
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.552594
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.654299
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.754648
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.856322
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:34.958230
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:35.059017
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:35.159784
ALERT: DeleteFunctionAttack  High   2015-11-22 17:12:35.261563
EVENT: Current alarm status is: High       2015-11-22 17:13:02.406449
EVENT: Current alarm status is: Medium     2015-11-22 17:13:22.561993
EVENT: Current alarm status is: Low        2015-11-22 17:13:42.693547

Sniffer stopped by keystroke.

#### Final results ####
Time elapsed: 177.172 seconds.
Packets sniffed: 161
- IP packets: 137 Percent: 85.0931677019 %
---- TCP packets: 105 Percent: 65.2173913043 %
---- UDP packets: 42 Percent: 26.0869565217 %
DNP3 messages: 52

DNP3 Application Layer Function Codes:
READ : 9
WRITE : 1
ENABLE_UNSOLICITED : 1
DELETE_FILE : 30
RESPONSE : 11
```